UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA
INGENIERÍA INFORMÁTICA SUPERIOR

ColDes: Collaborative Design

Desarrollo de una herramienta web para el dibujo gráfico colaborativo

Autor: José Miguel Blanco García
Tutor: Sara Tena García
Cotutor: David Diez Cebollero

Agradecimientos

"Todo viaje, por largo que sea, empieza por un solo paso."
Lao Tse

Cuanto impresiona echar la vista atrás y ver lo mucho que ha pasado desde que di el paso que inició esta etapa de mi vida, una etapa que cierro junto con este proyecto de fin de carrera.

Han sido siete largos años, siete años en los que he vivido experiencias de todo tipo junto a personas a las cuales tengo mucho que agradecer y es por eso, que a todas ellas, les dedico este proyecto de fin de carrera.

No puedo dejar de pensar lo orgulloso que se sentiría mi abuelo si me viese en este preciso momento de mi vida, es una etapa que no llegó a ver empezar, pero sin duda, si llegué a dar el paso que la dio inicio fue, en parte, por su culpa. Por eso, no quiero olvidarme de él en este momento.

Por supuesto a mi abuela, por su cariño, por su apoyo y ánimo inagotable, por esos largos ratos de estudio que pasaba en su casa, por creer en mí en absolutamente todo momento, por todo esto y más, gracias.

A mis padres, porque sin ellos no sería lo que soy ahora mismo, porque siempre han estado ahí apoyándome, en los buenos y en los malos ratos, por saber siempre animarme cuando más lo he necesitado, por alegrarse por mis triunfos y por sufrir con mis fracasos, por todo esto y mucho más, gracias.

No por mencionarla más tarde es menos importante, sin duda, uno de mis mayores apoyos a lo largo de estos siete años… hemos reído y llorado juntos, hemos pasado momentos muy buenos y por desgracia malos también. Una persona que siempre, absolutamente siempre, me ha tendido la mano cuando me he caído y cuando más ganas tenía de dejarlo todo. Que me ha ayudado a levantarme, que me ha ayudado a seguir creyendo que con esfuerzo y tesón podía llegar donde yo quisiera. Mucha parte de lo que este logro supone, es gracias a ella. Por todo ello, gracias Luz, sé que contigo a mi lado puedo hacer todo aquello que me proponga.

Mención aparte merecen esos compañeros de carrera que tan buenos momentos me han hecho pasar (y cuántas horas de sueño me han robado, sobre todo el último año). No quiero dejarme a nadie sin mencionar, pues todos han sido muy importantes a lo largo de estos años porque sin ellos, no me cabe la menor duda, no estaría donde estoy ahora mismo. Aunque hay menciones especiales que he de realizar. A Cesar, Erick y David, compañeros de prácticas y amigos, por esos largos ratos haciendo prácticas en los pasillos del Sabatinni, cafetería y demás lugares; compañeros, amigos y unos verdaderos fenómenos.

*"A todos, muchas gracias, esto, es **vuestro**"*

Resumen

El presente proyecto se engloba en el contexto de la colaboración, el cual podemos definir brevemente como un proceso mediante el cual dos o más personas (u organizaciones) trabajan juntos para alcanzar unos objetivos compartidos, todo esto obtenido gracias al intercambio de conocimientos, al aprendizaje mutuo y a la creación de consensos entre los distintos participantes.

En este proyecto se intenta abordar la problemática presente en el diseño colaborativo, donde varios diseñadores interactúan sobre el diseño de algún tipo de elemento. Esta interacción a priori es sencilla, en lo que a comunicación nos referimos, cuando todos los participantes se encuentran en el mismo contexto y lugar, sin embargo, *¿Qué pasa cuando estas personas no se encuentran en el mismo contexto y lugar?* El sistema **ColDes** *(Collaborative Design)* se presenta para dar soporte a esta necesidad de colaboración distribuida.

En la actualidad, gracias a la aparición de la **Web2.0**, se ha fomentado la creación de herramientas que soportan colaboración distribuida, como aplicaciones de *videoconferencias*, *chat's*, *pizarras online*, *wikis's*, *moodle's*, etc… No obstante, las pizarras online son las herramientas más adecuadas para dar soporte al desarrollo de diseños gráficos colaborativos, entendiendo pizarra online como un "*espacio en blanco en el que dibujar y escribir mediante el uso del ratón (o pantalla táctil) como "tiza" para que otros puedan visualizarlo*".

En **ColDes**, los usuarios pueden colaborar de forma distribuida con otros usuarios mediante el uso de pizarras online en las diversas salas creadas y gestionadas por ellos mismos dentro de la aplicación, estableciendo de esta forma un entorno de trabajo distribuido para el trabajo colaborativo entre personas que se encuentran en lugares y contextos distintos, mediante el cual pueden poner en común las ideas que puedan tener y plasmarlas en un diseño gráfico, usando en ese proceso distintos tipos de participación entre ellos.

Abstract

This project addresses the collaboration context, which may be concisely defined as an either physical or virtual dimension. This dimension enables one or more people or organization to work together in order to attain certain common goals. The interchanging of knowledge as well as the common learning and the creation of consensuses between the different participants make all this feasible. Thus, this investigation tries to tackle the current problems regarding the collaboration design - several designers interact in designing some sort of element. Such an interaction process appears most of the times a priori uncomplicated when it comes to communication - i.e. if all participants are right at the same context and place. *However, what happens when all these people move to other contexts or places?* Hence is hereby presented the **CoIDes** system (*Collaborative Design*) - suggested as problem-solving for this necessity of distributed collaboration.

Nowadays the emergence of the Web 2.0 has fostered the creation of bearing distributed collaboration tools, such as videoconference applications, chats, online whiteboards, wikis, moodle platforms, etc. Nonetheless, online whiteboards are the most appropriated tools for giving support to the development of collaborative graphic designs – meaning here online whiteboard "*a blank to draw and write by using the mouse (or touch screen) as 'chalk' in order to be visualized by others*".

With **CoIDes** users can collaborate in a distributed way with other users through online whiteboards. These whiteboards are allocated in different rooms within the application created and managed by the users themselves. By doing this, a distributed work environment for the collaborative work between people in different places and contexts is established. This environment lets users share their ideas and give them expression through graphic design. Different types of participation would be used between users during this process.

Índice de contenidos

Índice de Figuras

Índice de tablas

Glosario de términos y acrónimos

En este apartado se recogen todos los términos y acrónimos usados a lo largo del documento cuyo significado pueda no quedar suficientemente claro a lo largo del mismo.

Acrónimo	Descripción
ColDes	*Collaborative Design*
RAE	*Real Academia Española*
RIA	*Rich Internet Application*
IVA	*Impuesto sobre el Valor Añadido*
HTML	*Hyper Text Markup Languaje*
W3C	*World Wide Web Consortium*
MXML	*Minimal XML*
XML	*eXtendible Markup Languaje*
IDE	*Integrated Development Environment*
VPS	*Virtual private server*
RUP	*Rational Unified Process*
RU-F	*Requisito de Usuario Funcional*
RU-NF	*Requisito de Usuario No Funcional*
CU	*Caso de Uso*
JDBC	*Java Database Connectivity*
POJO	*Plain Old Java Object*
DAO	*Data Access Object*
UML	*Unified Modeling Language*
SWF	*Shockwave Flash*
MVC	*Modelo-Vista-Controlador*
E/R	*Entidad/Relación*
HTTP	*Hypertext Transfer Protocol*
PR	*Prueba*
INC	*Incidencia*
MEJ	*Mejora*
CON	*Concepto*
MU	*Manual de Usuario*

Tabla 1: Acrónimos

Termino	Descripción
Wiki	*Concepto que se utiliza en el ámbito de Internet para nombrar a las páginas web cuyos contenidos pueden ser editados por múltiples usuarios a través de cualquier navegador.* *http://definicion.de/wiki/*
Moodle	*Paquete de software para la creación de cursos y sitios Web basados en Internet. Es un proyecto en desarrollo diseñado para dar soporte a un marco de educación social constructivista.* *http://es.scribd.com/doc/6012729/Que-es-Moodle*
SecondLife	*Mundo virtual en 3D con interacciones de multijugador.*
Codecs	*Pequeño añadido que se instala en un sistema para poder reproducir audio o video que ha sido codificado o comprimido.* *http://www.um.es/atica/gat/gat2/soluciones/que-son-los-codecs/*
e-learning	*Aprendizaje electrónico o virtual.*
Drag&Drop	*Es una expresión que se refiere a la acción de mover con el ratón objetos de una ventana a otra o entre partes de una misma ventana.*
Plug-in	*Aplicación que se relaciona con otra para aportarle una función nueva.*
Perfil en la aplicación	*Tipo de usuario que define las acciones que este puede realizar dentro del sistema.*
Función en la sala	*Tipo de participantes que puede tener una sala dentro del sistema. Según la función que un usuario tenga dentro de una sala, este podrá realizar unas u otras acciones dentro de la misma.*
OpenSource	*Término con el que se conoce al software distribuido y desarrollado libremente.*
BlazeDS	*Framework que permite comunicar fácilmente una aplicación back-end distribuida con una aplicación front-end desarrollada en Flex.*
MySQL	*Sistema de gestión de base de datos relacional, multihilo y multiusuario.*
Servlet	*Programas que se ejecutan en el servidor, realizando la función de una capa intermedia entre una petición proveniente de un cliente HTTP, y las aplicaciones del servidor, pudiendo utilizar toda la paquetería y potencialidades del lenguaje.* *http://mundopc.net/%C2%BFque-son-los-servlet/*
Tooltip	*Herramienta visual que funciona al situar el ratón sobre algún elemento gráfico mostrando una ayuda adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra.*
Prototipo	*Representación limitada de un producto, permite a las partes probarlo en situaciones reales o explorar su uso, creando así un proceso de diseño de iteración que genera calidad.* *http://albertolacalle.com/hci_prototipos.htm*
JBoss	*Servidor de aplicaciones J2EE de código abierto implementado en Java puro.*

Tabla 2: Términos

1. Introducción y Objetivos

El presente proyecto se engloba en el contexto de la colaboración. De forma general, tal y como define la *RAE*, colaborar es *"Trabajar con otra u otras personas en la realización de una obra"* **[RAE01]**. Visto de otra forma, podríamos decir que la colaboración es un proceso mediante el cual dos o más personas (u organizaciones) trabajan juntos para alcanzar unos objetivos compartidos, todo esto obtenido gracias al intercambio de conocimientos, al aprendizaje mutuo y a la creación de consensos entre los distintos participantes **[Lon95]**.

Otro concepto fundamental en este proyecto es el diseño. Aplicando el concepto de colaboración, al de diseño, nos encontramos con el diseño colaborativo **[BDI]**, entendido como una metodología que permite a varios diseñadores intervenir sobre el diseño de algún tipo de elemento. En este tipo de diseño, en ocasiones se le brinda la oportunidad al usuario final del producto de hacer sus apreciaciones sobre el diseño de tal forma que éste se ajuste de la mejor manera posible a sus necesidades reales. En gran parte, esta forma de diseño surge gracias a la creciente demanda de productos más eficientes.

En la siguiente figura se aprecia un modelo de diseño colaborativo, en el que cada uno de los participantes aporta y evalúa el proyecto desde su perspectiva. De esta manera el resultado final de este tipo de diseño es el fruto el aporte de muchos especialistas interaccionando entre sí.

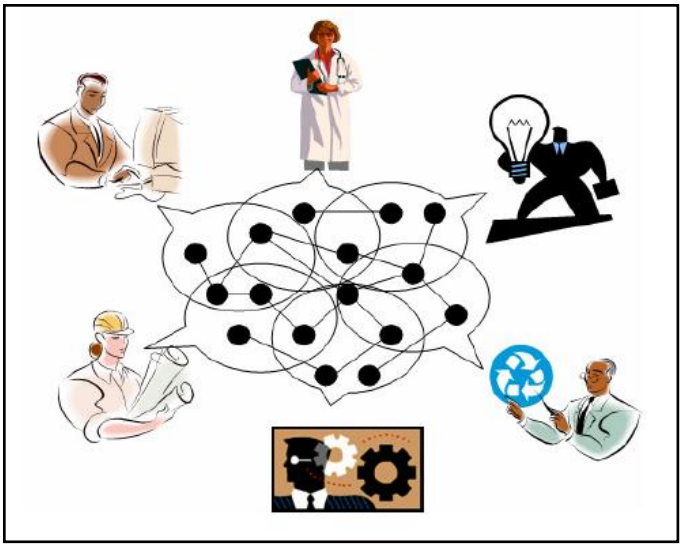


Figura 1: Modelo Conceptual de Diseño Colaborativo

1.1. Descripción del problema

La problemática principal que encontramos en ese tipo de entornos, en los que los participantes no se encuentran en el mismo entorno físico, es mediar entre ellos para facilitar la colaboración a la hora de diseñar.

En todo tipo de colaboración, es básica y necesaria la comunicación entre los distintos participantes para poder intercambiar conocimientos, aprender los unos de los otros y crear consensos entre ellos. La comunicación tiene como fin la interacción con otras personas, surge así la siguiente pregunta, *"¿Cómo interactuar con alguien que no se encuentra en el mismo lugar ni contexto?"*. Es aquí, donde entran en juego las nuevas tecnologías, mediante las cuales se pretende paliar las carencias que conlleva intentar establecer la comunicación con otras personas que no se encuentran ni en el mismo lugar ni contexto. De esta forma, nos centraremos en la colaboración mediada por la tecnología, también conocido como "*Technology-Mediated Collaboration*", donde la idea principal es usar las nuevas tecnologías para facilitar, alcanzar o soportar la colaboración entre personas (u organizaciones) **[Fad+03]**.

Los avances en los últimos años en las tecnologías centradas en la información y la comunicación en la red, han aportado nuevos modelos de resolución de problemas tanto para organizaciones como para personas. Como resultado, se ha comenzado a explotar los recursos virtuales de los que se disponen para la resolución de problemas de manera colaborativa, donde la colaboración entre los participantes cumple los siguientes requisitos **[Hay+10]**:

- Geográficamente dispersos.
- Vinculados a través de tecnología de la información y las comunicaciones.
- Colaboración a través del tiempo y el espacio.

En la actualidad, Gracias a la aparición de la **Web2.0**, se ha fomentado la creación de herramientas que soportan colaboración distribuida para diversos sistemas colaborativos de diseño **[She+08]**, como aplicaciones de *videoconferencias, chat's, pizarras online, wikis's, moodle's*, etc… No obstante, las pizarras online son las herramientas más adecuadas para dar soporte al desarrollo de diseños gráficos colaborativos, entendiendo pizarra online como un "espacio en blanco en el que dibujar y escribir mediante el uso del ratón (o pantalla táctil) como "tiza" para que otros puedan visualizarlo" **[OBOA]**. Existen ya aplicaciones que implementan este tipo de herramientas como por ejemplo: Scriblink, DabbleBoard o Cosketch.

La problemática identificada presente en todo caso las pizarras online, es el tipo de participación dentro de la misma a la hora de colaborar entre los distintos usuarios de la misma y la forma de controlar esta participación.

1.2. Objetivos

El objetivo fundamental que buscamos con la realización de este proyecto es desarrollar una pizarra online como una herramienta web que soporte de varias formas la participación distribuida de los usuarios en el modelado colaborativo de distintos diseños gráficos.

A continuación se definen los objetivos básicos que el sistema deberá cumplir:

- El sistema deberá dar soporte al usuario permitiéndole realizar diseños gráficos.
- Ha de permitir la comunicación entre los distintos usuarios de forma tal que se facilite la colaboración en la mayor medida posible.
- El sistema de comunicación y colaboración entre los usuarios ha de ser un sistema síncrono.
- Deberá permitir distintos tipos de participación, mediante los cuales se definan las políticas de colaboración entre los distintos usuarios.
- El sistema deberá ser multidiseño, permitiendo a un mismo usuario participar en distintos diseños en un mismo momento o en distintos momento.

1.3. Fases de desarrollo

En este apartado de la memoria se explican las distintas fases por las que ha pasado el desarrollo hasta llegar a su punto final.

Todo proyecto software se suele dividir en distintas etapas para facilitar así su gestión y control. El conjunto de las distintas etapas por las que va pasando el proyecto desde que se inicia hasta que se concluye, es lo que comúnmente se conoce como "*Ciclo de Vida de un Proyecto*". En la siguiente figura se exponen las distintas fases que se han realizado a lo largo de este proyecto:

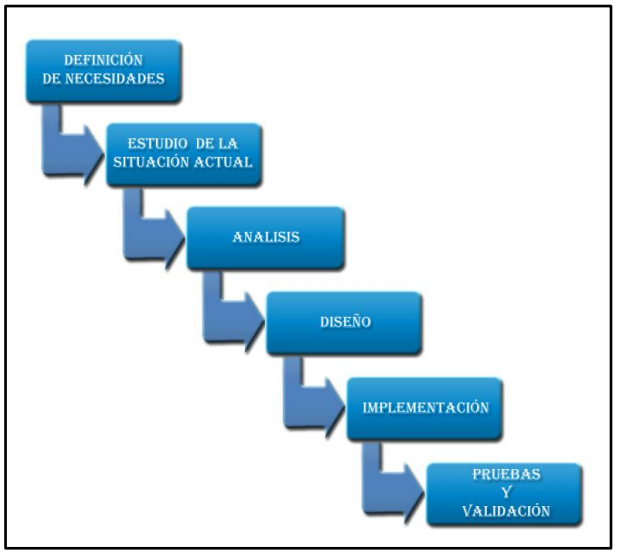


Figura 2: Ciclo de Vida del Software

- ***Definición de necesidades***: En esta fase es necesario definir las metas que se quieren obtener mediante la realización del proyecto. Para ello se definen una serie de objetivos fundamentales que la aplicación debe cumplir para cubrir las necesidades de los usuarios finales de la aplicación.
- ***Estudio de la situación actual***: En esta fase se realiza un estudio detallado de los entornos colaborativos y de las distintas tecnologías existentes capaces de dar soporte a este tipo de entornos. Con esto se busca dar forma al proyecto y seleccionar aquella tecnología que encaje mejor con las necesidades del mismo.
- ***Análisis***: En esta fase se determinan los distintos elementos que intervienen en el sistema que se quiere desarrollar, estructura, funcionalidades, requisitos y limitaciones del mismo.
- ***Diseño***: Una vez realizado el análisis, y tras saber que es lo que queremos conseguir al realizar este proyecto es necesario determinar el cómo se va a realizar. En esta fase se realiza una definición detallada de los distintos elementos del sistema: arquitectura, modelo de negocio, modelo de datos, interfaz de usuario, etc.
- ***Implementación***: En esta fase se es donde se comienza a codificar los distintos elementos del sistema (algoritmos, estructuras de datos, etc.) en el lenguaje especificado en la etapa anterior, cumpliendo todos las necesidades encontradas en la fase de análisis, y cumpliendo todos y cada uno de los requisitos obtenidos en la misma. Además en esta fase también incluiremos debugging, mediante el cual buscaremos garantizar que la implementación realizada del sistema no contiene errores de diseño o codificación.
- ***Pruebas y Validación***: En esta etapa se busca verificar que el sistema desarrollado cumple con los requerimientos expresados inicialmente por el cliente, en nuestro caso, buscaremos si los objetivos marcados en el apartado de objetivos se cumplen o no. Además, en esta fase se detectaran y solucionaran los posibles errores de la herramienta y se realizaran pequeñas mejoras solicitadas por el usuario para adaptar más el sistema a las necesidades del mismo.

1.4. Estructura de la memoria

En este punto de la memoria se presenta la estructura que sigue el presente trabajo:

- El **primer capítulo** presenta una breve introducción al proyecto, los objetivos que se pretenden alcanzar en la finalización del mismo, la estructura de la memoria y la metodología usada a lo largo del proyecto.
- En el **segundo apartado** se realiza el estado del arte, en el cual se ha realizado un estudio de los entornos colaborativos y de las tecnologías más apropiadas para dar soporte a este tipo de entornos.
- El **tercer apartado** describe la gestión realizada del proyecto a desarrollar, cubriendo los siguientes puntos: alcance del proyecto, plan de trabajo y gestión de recursos.
- En el **cuarto apartado** se expone la solución elegida para contemplar los objetivos fundamentales del proyecto. En este apartado se describe a fondo el

proceso de desarrollo que se seguirá a la hora de realizar el proyecto (análisis, diseño, pruebas, etc.).

- El **quinto apartado** presenta la evaluación del proyecto, donde se valida el plan de pruebas anteriormente presentado.
- El **sexto apartado** se exponen las conclusiones que se han obtenido tras la realización del proyecto, incluyendo las aportaciones realizadas, los problemas encontrados a lo largo de la realización del proyecto, líneas futuras para el sistema y opiniones personales del autor.
- El **séptimo apartado** se incluye toda la bibliografía usada para la realización del proyecto.

2. El estado del arte

A lo largo de este apartado se realizará una revisión de la situación actual de los entornos colaborativos, su presencia en la actualidad en la web y las distintas posibilidades que este tipo de entornos aportan a los usuarios. Además se realizara un análisis de dos tecnologías RIA, como soporte de la colaboración, Para ello se estudiarán sus características tecnológicas relevantes para la implementación del presente proyecto.

El desarrollo de un sistema que soporte la colaboración mediante la web supone un coste mayor mediante las aplicaciones web tradicionales, las cuales distan de cubrir los requisitos necesarios para soportar todas las necesidades de interacción entre las distintas partes que colaboran, frente a las aplicaciones RIA.

La base tecnológica de este proyecto consiste, en gran parte, en la comprensión y adecuación de las aplicaciones RIA, a las necesidades interactivas que éste plantea. Para ello es necesario conocer en qué estado se encuentran las aplicaciones y cuáles son las distintas tecnologías que se aplican en este dominio.

2.1. Entornos Colaborativos

Un entorno colaborativo, o "*collaborative workspace*" es una infraestructura o plataforma que permite a los usuarios trabajar juntos. Por ejemplo, para la recopilación de información, creación de contenidos de forma colaborativa o simplemente compartir datos entre ellos (en una *wiki*, *foro*, *chat*…). La principal característica de este tipo de software es la creación y modificación de contenido de forma colaborativa **[Bez+09]**.

Hablar de entorno colaborativo implica, en la mayoría de los casos, hablar de lo conocido como groupware, término usado para denotar aquellos productos o aplicaciones orientadas al soporte del trabajo en grupo. Los sistemas de tipo groupware ofrecen ventajas claras sobre los sistemas mono usuarios como **[Mar03]**:

- Facilitar la comunicación y habilitar la telecomunicación.
- Reunir múltiples perspectivas y formalismos.
- Formar grupos con un interés común.
- Ahorrar tiempo y costes en la coordinación.
- Facilitar la resolución de problemas en grupos.

Este tipo de entornos colaborativos abarcan desde editores de documentos compartidos, como *Google Docs*; sitios web abiertos y editables, como los *wikis*; incluso hasta redes sociales, que incluyen perfiles y herramientas de comunicación para dar un sentido de conexión y de comunidad, junto a herramientas para el trabajo colaborativo**[IBER10]**.

En la actualidad, gracias en gran medida a la expansión de Internet, este tipo de aplicaciones está completamente integradas en nuestro uso diario (email, mensajería instantánea, videoconferencias,…)

Para finalizar esta parte de la revisión veremos en los siguientes puntos algunos ejemplos de entorno colaborativo.

Voicethread

Voicethread (*http://voicethread.com/*) es una herramienta que permite crear álbumes multimedia en los que se pueden insertar documentos, imágenes, audio y video, con el valor añadido de dejar comentarios en formato de audio, video o texto, pudiendo además dibujar sobre los documentos.

Esta herramienta nos proporciona un entorno mediante el cual podemos mantener conversaciones alrededor de los elementos multimedia que deseamos, con las personas que queramos y pudiendo modificar el contenido de estos elementos en tiempo real. Mientras el resto de los participantes pueden ir observando los cambios.

En la siguiente figura se muestra un ejemplo de colaboración mediante el uso de de esta herramienta, donde se puede observar cómo los distintos participantes están colaborando y comentando el contenido mediante videoconferencia:

Figura 3: Ejemplo de interacción VoiceThread

De igual forma, en la siguiente imagen observamos cómo sobre otro elemento multimedia, algunos usuarios han aportado su opinión mediante comentarios en formato texto, y modificando el propio elemento compartido.

Figura 4: Comentario de foto en VoiceThread

MindMeister

MindMeister (*http://mindmeister.com/*) es una herramienta con la cual los usuarios pueden crear y editar en grupo documentos gráficos de forma online, como por ejemplo, mapas mentales. Usando las características para colaboración en tiempo real, permite sesiones globales de lluvia de ideas donde los usuarios pueden interactuar entre ellos en cualquier momento y lugar, sobre un mismo diagrama.

En la siguiente figura se muestra un ejemplo de un mapa mental creado mediante dicha aplicación. Este mapa mental se podría compartir con más personas y, éstas podrían acceder a él para comentarlo, aportar nuevas ideas, modificarlo, etc.

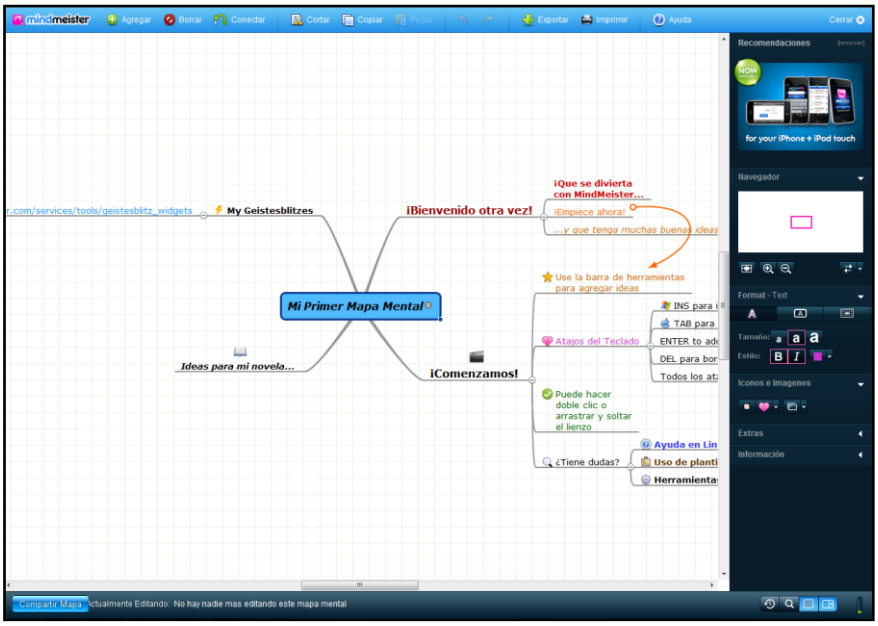


Figura 5: Ejemplo de mapa mental de MindMeister

Goolgle Docs

Google Docs (*https://docs.google.com/*) es una aplicación gratuita de Google que permite crear documentos en línea con la posibilidad de colaborar en grupo. Incluye un procesador de textos, una hoja de cálculo, programa de presentación básico y un editor de formularios destinados a encuestas.

Google Docs permite a sus usuarios:

- Crear y administrar documentos, hojas de cálculo y presentaciones online.
- Compartir y colaborar en tiempo real.
- Almacenamiento y organización del trabajo de los usuarios de forma segura.
- Control sobre los distintos permisos de los documentos del usuario, así como la opción de seleccionar los usuarios con quien compartirlos.

Google Docs ofrece compatibilidad con formatos de archivo tradicionales, permitiendo importar, editar en colaboración o publicar archivos .doc, .xls, .csv, .ppt, .txt, .html, .pdf y otros formatos.

Permite, además, acceso rápido a todos y cada uno de los documentos de los que dispone el usuario, tanto creados por él mismo, como compartidos con otros usuarios, tal y como se ve en la siguiente imagen:

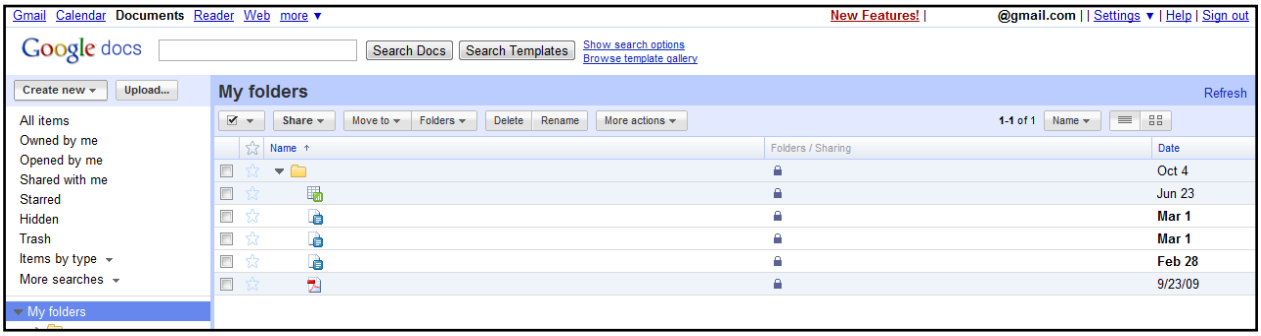


Figura 6: Pantalla principal de gestión de Google Docs

Acrobat

Esta propuesta de Adobe, *Acrobat* (*https://acrobat.com/*) tiene como objetivo fundamental permitir a los usuarios crear entornos colaborativos con funcionalidades propias. Es un conjunto de servicios on-line que incluye intercambio y almacenamiento de archivos, un conversor de documentos a PDF, procesador de textos y videoconferencia.

En la siguiente imagen vemos representado el concepto de espacio de trabajo que introduce Acrobat, donde el usuario puede introducir distintos tipos de documentos y compartirlos con otros usuarios de la aplicación **[ACRO10]**.

Figura 7: Espacio de trabajo en línea Acrobat

2.2. Tecnologías RIA

Las aplicaciones RIA son un tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Surgen como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales de escritorio. Las aplicaciones RIA son el resultado del avance tecnológico de hoy en día, que nos permite obtener aplicaciones con mayor capacidad de respuesta e interfaces graficas más avanzadas y cercanas a las aplicaciones tradicionales de escritorio **[Dei+08]**.

Los principales atractivos de este tipo de tecnologías son: la capacidad de incorporar interfaces de usuario intuitivas e interactivas en el cliente y la capacidad de estar disponibles para cualquier usuario en cualquier lugar **[Nas05]**.

Algunos de los beneficios que se pueden encontrar a la hora de desarrollar este tipo de aplicaciones son **[RIACT]**:

- Mejora de la experiencia visual del usuario, con unas interfaces mucho más atractivas visualmente hablando.
- En la mayoría de los casos, no es necesario instalar complementos para empezar a usarlas, o el proceso de preparación del entorno para su posterior uso es mínimo.
- En el momento en que se realizan nuevas versiones la actualización de las aplicaciones es totalmente transparente al usuario y se realiza de forma centralizada.
- El uso de las mismas se puede realizar desde cualquier tipo de ordenador (sin importar el sistema operativo que se utilice) y desde cualquier punto del planeta.

Partiendo del conocimiento a grandes rasgos de lo que son este tipo de aplicaciones y de los beneficios que ofrecen (visto en puntos anteriores), nos centraremos en el estudio de la nueva versión de HTML (HTML5) y las distintas posibilidades que presenta frente a las características aportadas por Adobe Flex, ya un clásico de las aplicaciones RIA.

Finalmente se realizará una comparativa entre ambas tecnologías que determine la tecnología a utilizar en el desarrollo del presente proyecto. Este análisis se realizará en base a los siguientes criterios:

- *Portabilidad*: Capacidad de la tecnología que permite a un programa o sistema ejecutarse en diferentes plataformas o arquitecturas con las mínimas modificaciones.
- *Dependencias*: Necesidades directas de la tecnología (aplicaciones, otros sistemas,…) para que el funcionamiento de los programas o sistemas sea el esperado.
- *Costes*: Costes implícitos en el uso de la tecnología como base para el desarrollo del proyecto.
- *Aceptación*: Indica si la tecnología está integrada hoy en día en el tipo de sistema que vamos a desarrollar.

- *Entornos de desarrollo*: Existencia de entornos de desarrollo para estas tecnologías, que faciliten en desarrollo de programas o sistemas basados en ellas.
- *Documentación*: Existencia de información que facilite el desarrollo de aplicaciones sobre las tecnologías estudiadas.

HTML5

HTML (*Hyper Text Markup Languaje*) es el formato estándar de la mayor parte del contenido actual de la Web, usado para describir la escritura y el contenido de una página en forma de texto. Es un lenguaje de marcado, que en la mayor parte de su uso ha sido limitado única y exclusivamente a texto estático y a imágenes, hasta la llegada de la nueva revisión de este estándar de la *W3C*: **HTML5**.

Esta nueva revisión del estándar estable una serie de nuevos elementos y atributos mediantes los cuales se intenta reflejar los usos típicos de los sitios web modernos, como por ejemplo la nueva etiqueta <nav>, que indica el bloque de navegación del sitio web.

De forma algo más genérica, las mejoras introducidas en esta nueva versión del lenguaje se pueden resumir en:

- Incorporación de nuevas etiquetas con codecs para mostrar contenido multimedia.
- Etiquetas para el manejo de grandes conjuntos de datos.
- Mejoras en los formularios e inclusión de nuevos tipos de datos que complementen las necesidades actuales (como por ejemplo, email, url, datetime…).
- Inclusión de tecnologías de Drag&Drop que permiten al usuario arrastrar elementos externos dentro de la web, como por ejemplo imágenes.
- Etiquetado especifico para manejo de web semántica (*Web 3.0*).

Uno de los elementos más interesantes de esta nueva versión del estándar es el *canvas*, el cual puede ser usado para la representación de gráficos, juegos u otro tipo de elementos visuales, todo ello sin necesidad de dotar al navegador de ningún tipo de componente adicional.

Hoy en día existen numerosas aplicaciones en HTML5 (junto con *JavaScrip*) que exprimen al máximo las funcionalidades de este nuevo componente del lenguaje de marcado. Desde aplicaciones para visualización de datos hasta juegos, pasando por aplicaciones para realizar dibujos sobre el *canvas*.

Un claro ejemplo de aplicación desarrollada en HTML5, y que explota al máximo las posibilidades de este nuevo elemento es *Sketchpad*:

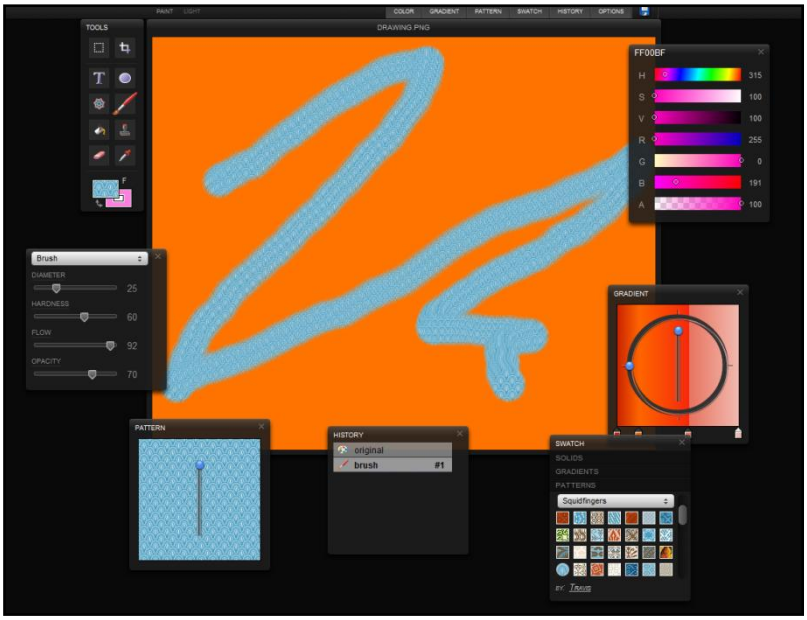


Figura 8: Sketchpad

Esta aplicación es similar a otras aplicaciones de dibujo existentes que funcionan directamente en el navegador web. La diferencia reside en el lenguaje de desarrollo, HTML5, y en que no recurre a tecnologías como Flash, que requieren de la instalación de componentes adicionales.

Otro de los usos actuales de este componente de HTML5, es la implementación de juegos, que, aunque muy lejos del potencial gráfico que presentan las tecnologías Flash, aportan un consumo mucho menor de recursos y sin necesidad de tener plugins instalados.

Figura 9: Mario Kart HTML5

No obstante, es un estándar que aún está en desarrollo, por lo que muchos de los elementos de los cuales dispone dependen de forma directa del navegador en el cual se ejecute la aplicación. En la siguiente imagen se muestra la compatibilidad actual, y previsible, de cada uno de los navegadores actuales con cada uno de los elementos más innovadores de HTML5 **[FOCUS]**:

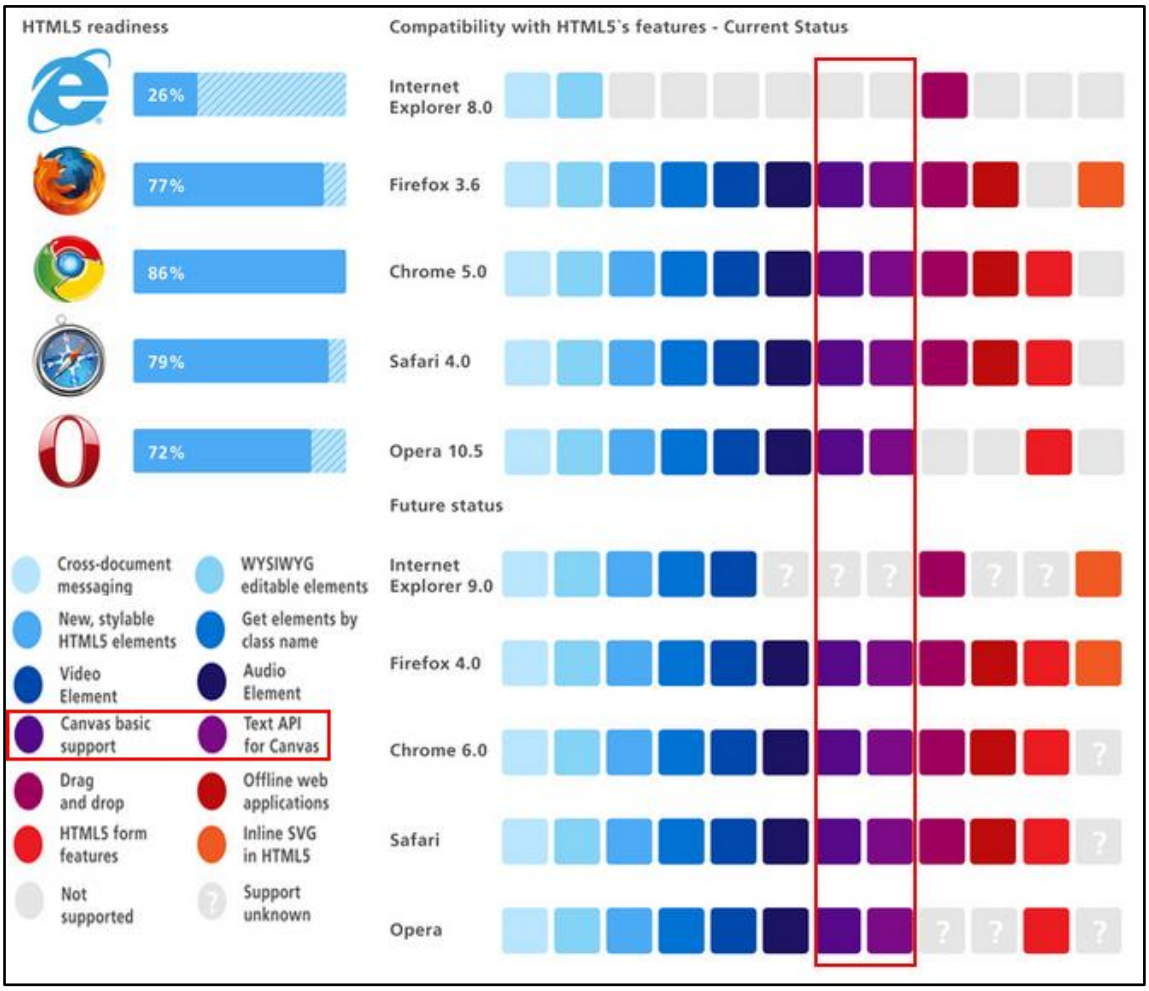
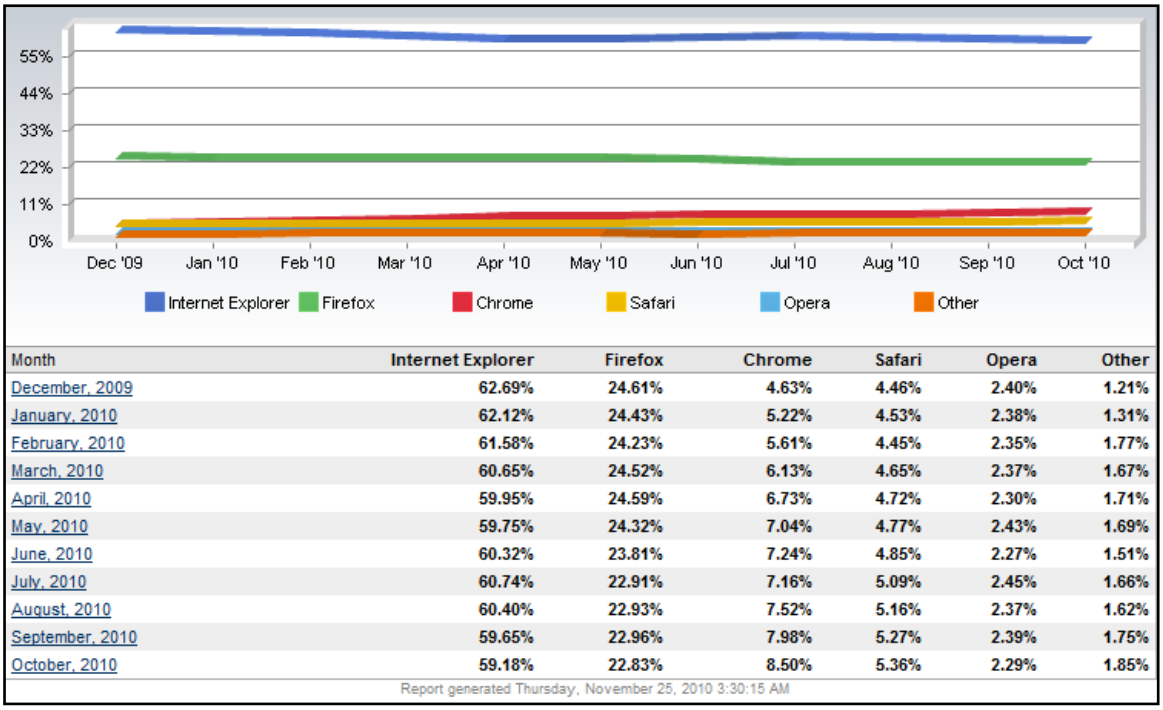


Figura 10: HTML5 vs Navegadores

Como se ve en la ilustración, los navegadores soportan la mayor parte de las mejoras que presenta la nueva revisión de HTML, con excepción de Internet Explorer, que ni en la versión actual del navegador ni para versiones futuras las soporta. Este es un detalle importante, pues Internet Explorer es uno de los navegadores más usados por los usuarios de la Web **[MARK]**:



Month	Internet Explorer	Firefox	Chrome	Safari	Opera	Other
December, 2009	62.69%	24.61%	4.63%	4.46%	2.40%	1.21%
January, 2010	62.12%	24.43%	5.22%	4.53%	2.38%	1.31%
February, 2010	61.58%	24.23%	5.61%	4.45%	2.35%	1.77%
March, 2010	60.65%	24.52%	6.13%	4.65%	2.37%	1.67%
April, 2010	59.95%	24.59%	6.73%	4.72%	2.30%	1.71%
May, 2010	59.75%	24.32%	7.04%	4.77%	2.43%	1.69%
June, 2010	60.32%	23.81%	7.24%	4.85%	2.27%	1.51%
July, 2010	60.74%	22.91%	7.16%	5.09%	2.45%	1.66%
August, 2010	60.40%	22.93%	7.52%	5.16%	2.37%	1.62%
September, 2010	59.65%	22.96%	7.98%	5.27%	2.39%	1.75%
October, 2010	59.18%	22.83%	8.50%	5.36%	2.29%	1.85%

Report generated Thursday, November 25, 2010 3:30:15 AM

Figura 11: Evolución del uso de los navegadores en el año 2010

Si buscamos una aplicación que sea portable a distintas arquitecturas, sin importar dónde y cómo se ejecute, necesitamos que cualquier navegador sea capaz de soportar la tecnología en la cual se basa nuestra aplicación, en este caso HTML5, y más en particular su nuevo elemento: *canvas*.

Esta nueva revisión puede llegar a librarnos de los *runtimes* propietarios como Flash, ya que es un estándar gratuito, lo que supone una clara ventaja para desarrollos Web. Todo ello unido a que el entorno de desarrollo de estas aplicaciones se puede basar única y exclusivamente en un procesador de textos planos, simples como el *notepad*, o más avanzados y con más funcionales como *ultraedit* o *notepad++* (de pago o gratuitos).

HTML5 viene con mucha fuerza gracias a grandes empresas como Google y Microsoft que han apostado muy fuerte por esta nueva tecnología. No obstante, por ejemplo, *Youtube*, una de las valedoras de HTML5 a través de su reproductor con soporte experimental (www.youtube.com/html5), ha dado a conocer ciertas limitaciones para la implantación de HTML5 que suponen una defensa de la necesidad de Flash para ofrecer vídeos online **[BYOUT]**.

En lo referente a la documentación sobre esta nueva revisión, ya existen muchos tutoriales y sitios de información (W3C posee una detallada lista de novedades de esta nueva revisión de HTML), pero aun esta algo incompleta y aún no es definitiva, dado que el estándar aún no está cerrado.

Adobe Flex

Este término engloba una serie de tecnologías creadas para dar soporte al despliegue y desarrollo de aplicaciones RIA, basadas en su plataforma Flash.

En su origen, las tecnologías Flash fueron pensadas principalmente para la creación de interfaces de usuarios. Flex ofrece un lenguaje basado en estándares y un modelo de programación que admite los patrones de diseño habituales. Para ello incluye:

- *MXML*, un lenguaje declarativo basado en XML, se utiliza para describir el aspecto y comportamiento de la interfaz de usuario.
- *ActionScript*, un lenguaje de programación orientado a objetos que se utiliza para crear la lógica de clientes.
- *Biblioteca de componentes*: muy completa con más de 100 componentes de interfaz de usuario extensibles y de eficacia demostrada para crear aplicaciones de Internet sofisticadas (RIA). En la siguiente figura se ven los distintos componentes que disponemos en la versión 3 de Flex.

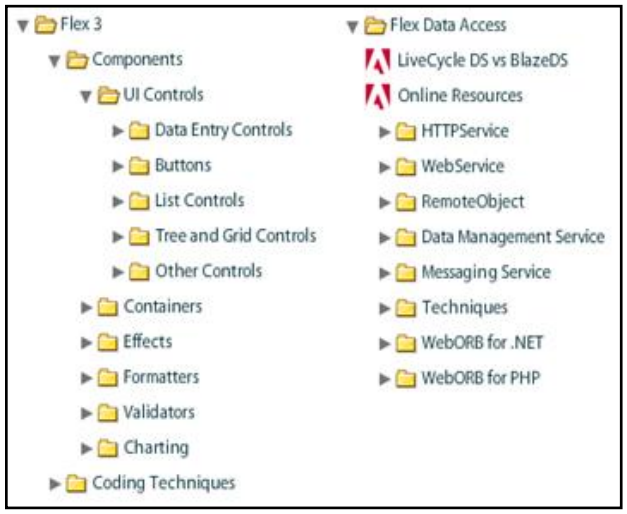


Figura 12: Componentes Adobe Flex

Otras características que ofrece Flex, y que resultan atractivas son las siguientes:

- Entorno de programación familiar a los desarrolladores con conocimientos sobre programación orientada a objetos.
- Sintaxis de definición formal de clases.
- Estructura formal de paquetes y herencia.
- Tipado de variable, parámetros y valores de retorno.
- Funciones get/set implícitas.
- Introducción al concepto de clase cerrada.

En la siguiente ilustración se muestra de forma genérica la estructura que siguen las aplicaciones que se basan en este tipo de tecnologías:

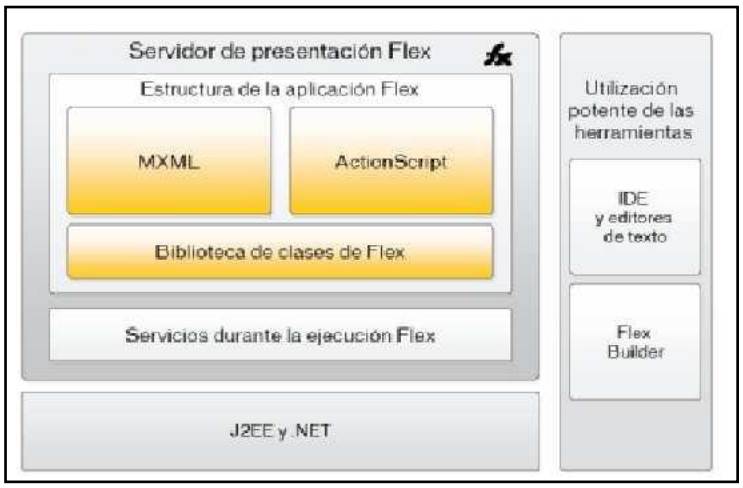


Figura 13: Arquitectura aplicación Flex

Las aplicaciones creadas con esta tecnología hacen uso de Adobe Flash Player, o para entornos de escritorio, Adobe AIR. Este punto en principio podría ser un problema, ya que en ausencia de estos elementos las aplicaciones creadas bajo esta tecnología no funcionarían.

Sin embargo, Adobe Flash está instalado en aproximadamente el 99% por ciento de los ordenadores conectados a internet, pues muchas aplicaciones web de la red requieren de este elemento para la correcta visualización de su contenido (como por ejemplo *YouTube*, uno de los portales más visitados de internet) **[ADOBW]**:

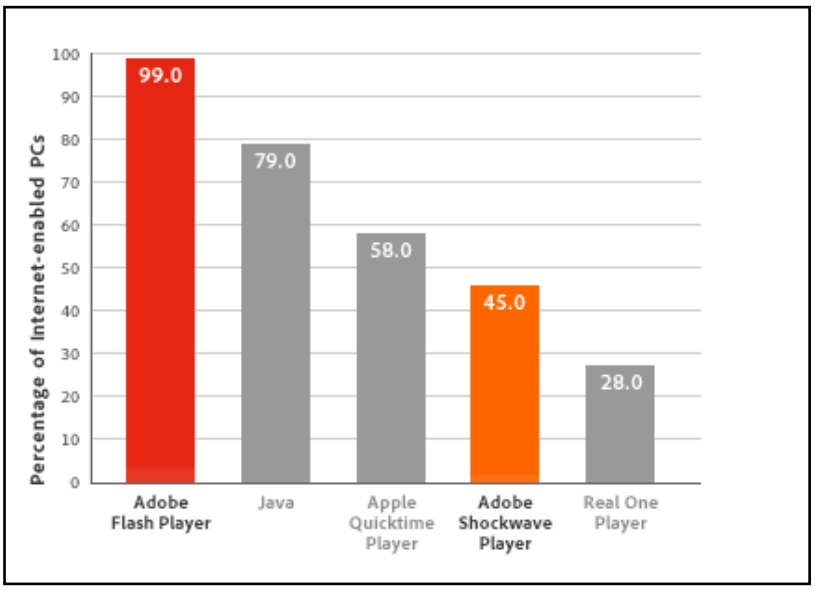


Figura 14: Uso de Adobe Flash Player

Aún con este dato, la dependencia de estas aplicaciones a un soporte Flash se puede traducir en algún que otro problema:

- Fallos de la plataforma Flash provocaría fallos en la aplicación de forma directa.
- La carga de los elementos puede resultar muy pesada, y dependerá en gran medida del ancho de banda del usuario final de la aplicación.
- Los contenidos de las aplicaciones web que se han desarrollado bajo esta tecnología son contenidos Flash, los cuales no son indexados por los buscadores.

Para este tipo de aplicaciones existen diversos IDE's (entornos de desarrollo) que facilitan en gran medida el desarrollo de estas: *Adoble Flash Builder*, cuyo propietario es *Adobe* y es de pago, un plug-in para el IDE *Eclipse*, que proporciona prácticamente las mismas funcionalidades que el IDE de *Adobe*, e incluso es posible desarrollar aplicaciones en *Flex* únicamente con un editor de texto y el SDK (gratuito) apropiado para la compilación de los archivos fuentes.

Las aplicaciones RIA que se basan en la tecnología Flash, son usadas por un gran número de empresas. En la siguiente figura podemos ver un ejemplo de algunas empresas **[**CYCLW**]**:

Figura 15: Ejemplo de empresas que usan Flex

Un ejemplo claro del uso de este tipo tecnologías llevada al ámbito que este proyecto quiere tratar, los entornos colaborativos, y más concretamente, al campo de las pizarras online, es la siguiente aplicación, *Dabbleboard*:

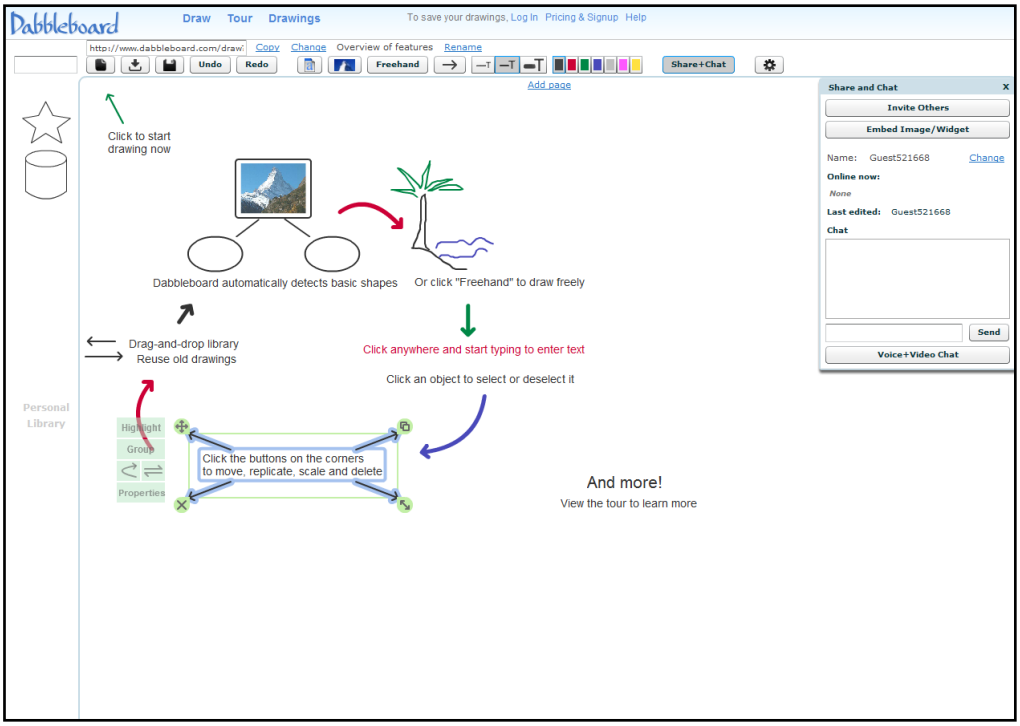


Figura 16: Dabbleboard

Dabbleboard es una pizarra online colaborativa que permite al usuario dibujar o exponer todo tipo de diseños. Ofrece al usuario una interfaz realmente sencilla e intuitiva, mediante la cual se permite a los usuarios de la aplicación, insertar imágenes, editar sus propiedades, dibujar todo tipo de elementos, realizar trazados a mano alzada, chatear con contactos que están editando sobre la misma pizarra, guardar el trabajo realizado, exportarlo a imagen al propio ordenador desde el cual el usuario está trabajando, etc...**[DABBL]**.

Finalmente, en lo referente a la documentación, existe infinidad de documentación (oficial de Adobe y no oficial) sobre esta tecnología, tanto en soporte digital como en papel y tutoriales con ejemplos guiados (ej. Tour de Flex).

Conclusión

Como ya se ha podido apreciar en las anteriores secciones de este apartado, tanto *Flex* como *HTML5*, poseen las características necesarias para la implementación del proyecto, sin embargo, llegados a este punto es interesante analizar y valorar las posibilidades que cada una de las tecnologías ofrece para llegar a una conclusión final.

Un criterio de decisión que también se valorara una vez realizada la revisión de las distintas tecnologías será los propios conocimientos de los que se disponen de cada una de ellas.

En la siguiente tabla, se muestran los criterios que se han considerado esenciales para la elección de una de las tecnologías que hemos estudiado a lo largo de los anteriores puntos.

Criterio	Adobe Flex (Flash)	HTML5
Portabilidad	Muy buena	Buena
Dependencias	Alta Requiere Adobe Flash o Adobe Air dependiendo del formato de la aplicación.	Ninguna. Únicamente la compatibilidad del explorador donde se ejecute
Costes	Medio	Ninguno
Aceptación	Excelente	Buena/Prometedora
Entornos de desarrollo	Buena Flex Builder Plug-in de eclipse	Buena Con cualquier editor de texto Más laborioso
Documentación	Excelente	Muy buena, pero incompleta
Especificación/SDK	Open Source (MPL)	Open Standard (WHATWG)
Conocimientos propios	Buena	Escasa

Tabla 3: Criterios Tecnologías

Por todo ello, el desarrollo de la aplicación finalmente se realizara en Flex y en Java. Usaremos Flex para la parte visual de la aplicación y Java, como soporte para la parte que se centra en el control de datos de la misma (control de acceso, guardado de datos en base de datos…).

3. Gestión del proyecto software

La gestión de proyecto es una actividad esencial en todo desarrollo software. Esta actividad permite organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definidos. Así pues, en este punto se detallarán las estimaciones de tareas, recursos (tanto humanos como materiales) y presupuesto.

Por otro lado, se ha optado por realizar la gestión de proyecto como si se tratase de la gestión de un proyecto software real. Por esta razón, todas las estimaciones serán llevadas a cabo de una forma simulada en el que se podrá contar con todos los recursos, tanto humanos como materiales, que se estimen necesarios para realizar el proyecto. Además, todo presupuesto realista tiene como objetivo una rentabilidad, para ello se sumarán los márgenes de beneficio y riesgo a la oferta, de forma que se asegure su rentabilidad.

3.1. Alcance del proyecto

La idea de este proyecto es la de dar soporte a la colaboración distribuida. Por ello se ha optado por desarrollar una pizarra colaborativa, mediante la cual los distintos usuarios de la aplicación puedan poner en común sus ideas de la forma más sencilla posible.

El sistema se plantea como una herramienta web colaborativa, que sirve como soporte para la participación distribuida de los usuarios en el modelado de diseño grafico mediante el uso de pizarras online.

El soporte que dará el sistema a los usuarios del mismo se basará en la creación de salas, que tendrán una pizarra y un chat, mediante los cuales los usuarios podrán colaborar siguiendo un tipo de participación específico de la sala. Por otro lado, el sistema permitirá entre otras funciones:

- Gestión de usuarios (altas, bajas y modificaciones) para usuarios administradores.
- Gestión de salas (modificaciones y gestión de usuarios participantes) para usuarios administradores.
- Gestión de los datos personales del usuario.
- Gestión de las salas propias del usuario (altas, bajas y modificaciones) y de los usuarios presentes en las mismas (invitaciones y expulsiones).

3.2. Plan de trabajo

El proyecto se dividirá en las siguientes fases o tareas: análisis, diseño, implementación y pruebas. De forma adicional, para tener un mayor conocimiento de la situación actual de los entornos colaborativos, se realizara una fase previa en la que se estudiaran las distintas opciones que se tienen para llevar a cabo el proyecto, distintas tecnologías, ejemplos ya presentes en el mercado, etc.

En la siguiente tabla se resumen las tareas previstas para el proyecto y las personas asociadas a dichas tareas:

Fase	Personas
Estudio previo	Jefe de Proyecto
	Analista
Análisis	Jefe de Proyecto
	Analista
Diseño	Jefe de Proyecto
	Diseñador/Programador
Implementación	Diseñador/Programador
Pruebas	Jefe de Proyecto
	Tester

Tabla 4: Tareas - Recursos

Para llevar a cabo las distintas tareas de la forma más eficiente posible, es necesario realizar una planificación mediante la cual organizar el tiempo y los recursos de los cuales disponemos.

A continuación se muestra el cronograma con las distintas fases del proyecto y sus tareas correspondientes, así como el tiempo estimado para realizar cada una de estas. La elaboración de esta planificación se ha adecuado a las siguientes premisas:

- Jornada laboral de 8 horas diarias.
- Semana laboral de 5 días (lunes a viernes).

Tarea		Duración	Fecha de inicio	Fecha de fin
Estudio preliminar	Estudio previo	20 días	01/11/2010	29/11/2010
	Esbozo de la solución	5 días	29/11/2010	06/12/2010
	TOTAL	***25 días***	***01/11/2010***	***06/12/2010***
Análisis	Definición de requisitos	5 días	07/12/2011	14/12/2010
	Especificación de requisitos	25 días	15/12/2010	31/01/2011
	TOTAL	***30 días***	***07/12/2011***	***31/01/2011***
Diseño	Diseño modelo de datos	14 días	01/02/2011	21/02/2011
	Diseño lógica de negocio	14 días	22/02/2011	11/03/2011
	Diseño interfaz de usuario	25 días	14/03/2011	15/04/2011
	TOTAL	***53 días***	***01/02/2011***	***15/04/2011***
Implementación	Implementación	94 días	25/04/2011	30/08/2011
	TOTAL	***94 días***	***25/04/2011***	***30/08/2011***
Pruebas	Pruebas	25 días	05/09/2011	07/10/2011
	TOTAL	***25 días***	***05/09/2011***	***07/10/2011***

Tabla 5: Planificación inicial del desarrollo

Como se puede observar en la planificación, el proyecto tendrá comienzo el día 1 de Noviembre de 2010 y tiene como fecha prevista de finalización el 7 de Octubre de 2011, siendo por tanto la duración del proyecto de once meses. No obstante, teniendo en cuenta que las planificaciones de proyecto pueden sufrir variaciones a lo largo del desarrollo, se opta por dar una duración total al proyecto de doce meses.

3.3. Gestión de recursos

En este apartado presentaremos una estimación detallada de los costes de los distintos recursos necesarios para la realización del proyecto. Para presentar los costes de la forma más clara posible, se han dividido los recursos en distintas categorías según la naturaleza de los mismos:

- Costes de personal.
- Costes de equipos.

3.3.1. Costes de personal

Al tratarse de un proyecto software, serán fundamentales los recursos humanos. En este sentido es imprescindible contar con un equipo de trabajo que sea capaz de entender el dominio del problema y además se encargue de proponer soluciones que cumplan los requerimientos del cliente. En este equipo de trabajo se necesitará ocupar los siguientes roles:

- ***Jefe de proyecto***: Encargado de coordinar, dirigir y controlar al resto de las personas participantes del proyecto.
- ***Analista***: Responsable de investigar, planear, coordinar y recomendar opciones tecnológicas para cumplir con los requerimientos de la empresa.

- ***Diseñador***: Encargado de diseñar posibles soluciones al problema que se plantea, bajo la supervisión del jefe de proyecto.
- ***Programador***: Encargado de trasladar las especificaciones del analista y diseñador al código de la tecnología seleccionada.
- ***Tester***: Responsable de dirigir el proceso de pruebas en el que se busca verificar que el software cumple con los requisitos y la funcionalidad establecida.

Asumiendo la poca envergadura del proyecto, no se contara con un especialista en cada rol. Se considera razonable contar con un equipo de cuatro personas para llevar a cabo la realización del proyecto, donde cada una de ellas deberá ocupar a lo largo de la duración del proyecto cada uno de los roles definidos anteriormente. De esta forma finalmente tendremos cuatro personas que ocuparan de la siguiente forma los roles:

- ***Jefe de proyecto***
- ***Analista***
- ***Diseñador/Programador***
- ***Tester***

De esta forma, en la siguiente tabla se detalla la estimación del salario bruto mensual de cada uno de los trabajadores participantes del proyecto. Por último, se ha considerado que no se necesitará que cada recurso tenga dedicación completa a este proyecto. A partir de la dedicación estimada de cada recurso a cada tarea, se ha planificado el reparto de la jornada laboral.

Bruto Mensual = (€/dia)*(horas/dia)*(días/mes)

Cargo	€/Hora Mínimo	€/Hora	Horas/Día	Días/Mes	Bruto mensual
Jefe de Proyecto	6,30 €	25,40 €	2	22	1117,6 €
Analista	6,30 €	20,75 €	4	22	1826 €
Diseñador/Programador	6,30 €	20,75 €	5	22	2282,5 €
Tester	5,22 €	20,75 €	3	22	1369,5 €
				TOTAL	***6595,6 €***

Tabla 6: Estimación salario bruto de personal

Tras estimar el coste mensual del personal, es necesario calcular las cuotas de cotización a la seguridad social. Para ello se ha obtenido la actualizada de las bases de cotización actuales se han consultado los datos que aporta el Ministerio de Trabajo e Inmigración español [COTSEG]:

Grupo de Cotización	Categorías Profesiones	Bases mínimas	Bases máximas
1	Ingenieros y Licenciados. Personal de alta dirección no incluido en el articulo 1.3.c del Estatuto de los Trabajadores	1045,20 €/mes	3230,10 €/mes
2	Ingenieros Técnicos, Peritos y Ayudantes Titulados	837.00 €/mes	3230,10 €/mes

Tabla 7: Bases de Cotización de 2011

Otro punto que es necesario analizar es el sistema fiscal y tributario español, donde la empresa contratante ha de asumir el pago de un porcentaje de la cotización al sistema de seguridad social publico de cada empleado. En la siguiente tabla presenta los porcentajes de cotización a la seguridad social asumibles por la empresa y por los propios trabajadores.

Contingencias	Empresa	Trabajadores	Total
Comunes	23,6%	4,7%	28,3%
Horas Extraordinarias	12%	4,7%	14,7%
Resto de Horas Extraordinarias	23,6%	4,7%	28,3%

Tabla 8: Tipos de cotización (%)

La cuota particular a ingresar es el resultado de aplicar a la base de cotización pertinente, el tipo o porcentaje de cada trabajador que se establece cada año.

Cuota a Ingresar = Base de Cotización x Tipo de Cotización / 100

Nombre	Bruto mensual	Base Mínima	Base Cotizada	Tipo	Cuota
Jefe de Proyecto	1117,6 €	1045,20 €	1117,6 €	23,6%	263,75 €
Analista	1826,0 €	837,00 €	1826,0 €	23,6%	430,94 €
Diseñador/Programador	2282,5 €	837,00 €	2282,5 €	23,6%	538,67 €
Tester	1369,5 €	837,00 €	1369,5 €	23,6%	323,20 €

Tabla 9: Cuota a ingresar en la seguridad social

De esta forma podemos estimar el coste total en lo que a personal se refiere en la siguiente tabla, teniendo en cuenta que el periodo total de desarrollo de la aplicación será de doce meses:

Nombre	Bruto mensual	Cuota	Meses	Coste Total
Jefe de Proyecto	1117,6 €	263,75 €	4	5525,40 €
Analista	1826,0 €	430,94 €	4	9027,76 €
Diseñador/Programador	2282,5 €	538,67 €	8	22569,36 €
Tester	1369,5 €	323,20 €	1	1692,7 €
			TOTAL	***38815,22 €***

Tabla 10: Coste final del personal

3.3.2. Costes de equipo

En esta sección se detallan tanto los costes asociados a los equipos informáticos necesarios para el desarrollo del proyecto, como los recursos software usados para su desarrollo.

Para la realización del proyecto se han necesitado los siguientes elementos.

- Ordenador mediante el cual el trabajador puede realizar las distintas tareas descritas anteriormente en el proyecto. Para ello se han adquirido cuatro portátil Asus *A53SJ-SX338V*.
- Ratón óptico para mejorar el manejo del ordenador personal del trabajador.
- Impresora para la impresión de la documentación necesaria requerida a lo largo del proyecto.
- Para montar un entorno de desarrollo, es imprescindible alquilar un servidor donde poder alojar un repositorio del código que se vaya generando a lo largo del desarrollo, montar el servidor donde ira desplegada la aplicación y los distintos prototipos de la misma para poder realizar pruebas durante el desarrollo. Por ello se ha seleccionado un *servidor VPS* con sistema operativo *Ubuntu 10.04 LTS* de 1024 MB de RAM [SERUB].
- Para la realización de toda la documentación del proyecto se ha optado por contratar una versión de *Microsoft Office 2010*, lo mas económica y completa posible [MICOF].
- Por último, para llevar un control del proyecto se ha elegido la herramienta *Microsoft Project 2010* [MICPR].

En la siguiente tabla vemos el desglose de precios de todos los elementos, donde el coste imputable se calcula según la siguiente fórmula:

Coste Imputable = A/B*C*D/100

Donde:
A = nº de meses desde la fecha de facturación en que el equipo es utilizado.
B = periodo de depreciación (60 meses).
C = coste del equipo (sin IVA).
D = % del uso que se dedica al proyecto (habitualmente 100%).

Descripción	Coste (€)	Unidades	Coste total (€)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Asus A53SJ-SX338V	579,00	4	2316,00	100%	12	60	463,20 €
Ratón M115 LOGITECH	11,01	4	44,04	100%	12	60	8,81 €
Impresora inalámbrica multifunción Dell V715w	136,00	1	136,00	100%	12	60	27,20 €
Servidor VPS Ubuntu 10.04 LTS	369,78	1	369,78	100%	7	60	43,14 €
Microsoft Office 2010: Office Hogar	249,00	4	996,00	100%	12	60	199,20 €
Microsoft Project 2010	1300,00	1	1300,00	100%	12	60	260,00 €
						TOTAL	***1.001,55 €***

Tabla 11: Costes de material software y hardware

3.3.3. Coste total del proyecto

Con todo lo expuesto en los puntos anteriores del documento, se obtiene el siguiente coste final.

Descripción	Coste Imputable
Personal	38815,22 €
Equipo Software y Hardware	1.001,55 €
TOTAL	***39816,77 €***

Tabla 12: Coste total asociado al proyecto

En la siguiente tabla se presenta el beneficio a obtener tras la realización del proyecto. Este beneficio se calcula en base a un margen de beneficio del 20%.

Coste Total	Margen de Beneficio	Beneficio
39816,77 €	20%	***7963,35 €***

Tabla 13: Calculo de beneficios del proyecto

Adicionalmente se aplica un margen de riesgo del 5%, sobre el coste total del proyecto.

Coste Total	Margen de Beneficio	Beneficio
39816,77 €	5%	***1990,84 €***

Tabla 14: Calculo de riesgos del proyecto

A continuación se presenta en la siguiente tabla el coste final del proyecto, aplicando el margen de beneficio y de riesgos calculados anteriormente, sin aplicar el IVA del 18%.

Descripción	Coste Imputable
Coste del proyecto	39816,77 €
Beneficio	7963,35 €
Riesgo	1990,84 €
TOTAL	***49770,96 €***

Tabla 15: Coste total asociado al proyecto

Para facilitar el pago de los costes del proyecto, se establecerán dos plazos de pago, uno al inicio y otro al final del proyecto. La siguiente tabla presenta el cálculo final incluyendo el IVA.

Cuota	Fecha	IVA aplicable	% del Coste	Parte del Coste (Sin IVA)	Total a pagar
Primera Cuota	Inicio del proyecto	18%	40%	19908,38 €	23491,89 €
Segunda Cuota	Fin del proyecto	18%	60%	29862,58 €	35237,84 €
				TOTAL	***58729,73 €***

De esta forma, el precio total, aplicando el IVA pertinente del 18%, seria de **CINCUENTA Y OCHO MIL SETECIENTOS VEINTINUEVE CON SETENTA Y TRES CENTIMOS** (***58729,73 €***).

4. Solución

En este apartado se explicará de forma detallada la solución llevada a cabo para resolver el problema descrito en los anteriores puntos del documento. De forma algo más específica, nos centraremos en el proceso de desarrollo seguido para llegar a la solución de la problemática que se planteaba.

4.1. Descripción de la solución

Como ya hemos comentado en anteriores puntos del documento, la problemática que se plantea es la colaboración entre personas en entornos distribuidos. De esta forma lo que buscamos en la solución del proyecto es diseñar una herramienta que de soporte a esta colaboración.

Así pues, para dar soporte a esta necesidad de colaboración distribuida, se ha optado por desarrollar una pizarra colaborativa, mediante la cual los distintos usuarios de la aplicación puedan poner en común sus ideas de la forma más sencilla posible.

El sistema se plantea como una herramienta web colaborativa, que sirve como soporte para la colaboración distribuida de los usuarios en el modelado de diseño grafico mediante el uso de pizarras online y chat para establecer comunicación entre los usuarios que participan en la pizarra.

El soporte que dará el sistema a los usuarios del mismo se basará en la creación de salas, que tendrán una pizarra y un chat, mediante los cuales los usuarios podrán colaborar. De forma algo más visual, la herramienta se regirá de la siguiente forma:

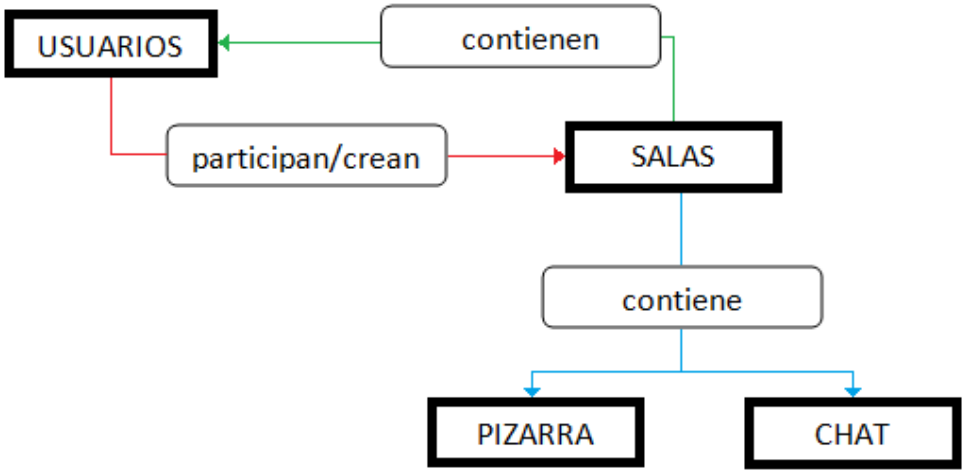


Figura 17: Esquema de Salas

La colaboración dentro de la sala podrá estar regida por turnos, o todos a la vez.

- En una sala en la que la participación se rige por turnos existe un elemento denominado pincel que indica que usuario de los participantes dentro de la sala puede darle uso a la pizarra para esbozar sus ideas. El pincel rotara entre los usuarios de la sala mediante peticiones realizadas por los mismos solicitando el uso del pincel. Esta rotación seguirá una política definida mediante la cual se seleccionara, entre todas las peticiones, el próximo usuario dueño del pincel. Por defecto, la política definida para gestionar las peticiones y la selección de próximo dueño del pincel se ha optado por implementar una política FIFO (*First Input First Output*), donde la primera petición que llega es la primera en obtener el pincel dentro de la sala.
- Por último, en las salas donde la colaboración es todos a la vez, todo usuario dentro de la sala tiene su pincel, por lo que pueden darle uso siempre que ellos lo deseen sin necesidad de solicitar el pincel para ello. Es importante remarcar, que este tipo de colaboración, puede llevar consigo problemas de sincronismo en el contenido del lienzo de cada uno de los usuarios.

Por otro lado, el sistema permitirá entre otras funciones:

- Gestión de usuarios (altas, bajas y modificaciones) para usuarios administradores.
- Gestión de salas (modificaciones y gestión de usuarios participantes) para usuarios administradores.
- Gestión de los datos personales del usuario.
- Gestión de las salas propias del usuario (altas, bajas y modificaciones) y de los usuarios presentes en las mismas (invitaciones y expulsiones).

4.2. El proceso de desarrollo

En este punto se determinará el proceso de desarrollo del sistema propuesto en la solución, siendo este proceso la base para que el proyecto se realice de forma correcta. En los siguientes apartados se expondrán las distintas fases de desarrollo llevadas a cabo para la conclusión del proyecto:

- *Análisis*
- *Diseño*
- *Implementación*
- *Pruebas y Validación*

4.2.1. Análisis

Este apartado tiene como meta final especificar las características y objetivos que el nuevo sistema deberá proporcionar así como su análisis y clasificación

4.2.2. Definición de requisitos

Dentro de los requisitos de sistemas hemos realizado la siguiente división para una mejor comprensión de los mismos.

- *Requisitos funcionales*: detallan la funcionalidad que el usuario demanda para cumplir sus objetivos.
- *Requisitos no funcionales*: definen la manera en la que el software debe estar construido (consideraciones hardware, imposiciones software, etc.).

Los atributos empleados para la especificación y diferenciación de los distintos requisitos son:

- Identificador: valor que diferenciará unívocamente a cada requisito.
  - Funcional (RU-FXXX)
  - No Funcional (RU-NFXXX).
- Descripción: explicación breve de lo que demanda el requisito.
- Necesidad: Importancia de la implementación o no del requisito en cuestión. Los posibles valores son: Esencial, Deseable u Opcional. Aquellos categorizados como “esencial” son obligatorios.
- Prioridad: Indica el orden en que deben implementarse. Puede ser: alta, media o baja. Los requisitos de “alta” prioridad deben crearse antes que los de prioridad “media”, y éstos, antes que los de prioridad “baja”.

Requisitos Funcionales

Identificador	Necesidad	Prioridad	Descripción
RU-F001	Esencial	Alta	La aplicación deberá dar soporte a la participación distribuida de los usuarios. La colaboración deberá realizarse mediante una pizarra online y un chat.
RU-F002	Esencial	Alta	El acceso a la aplicación se realizar mediante un componente de autenticación que pedirá al usuario un nombre y una contraseña.
RU-F003	Esencial	Alta	La aplicación deberá dar lo opción a los nuevos usuarios de registrarse en el sistema.
RU-F004	Esencial	Alta	A la hora de registrar un nuevo usuario el sistema pedirá los siguientes datos: • Nombre. • Apellidos. • Correo electrónico. • Nombre de usuario (username). • Contraseña (password).
RU-F005	Deseable	Media	Los usuarios registrados en el sistema, de forma inicial estarán desactivados hasta que un administrador del sistema de el visto bueno a este registro y lo active.
RU-F006	Esencial	Alta	La aplicación tendrá dos perfiles diferentes para los usuarios: administrador y diseñador.
RU-F007	Esencial	Alta	Los perfiles de la aplicación no son excluyentes. Podrán existir: administradores, diseñador y administradores- diseñador.
RU-F008	Deseable	Alta	La aplicación deberá soportar la gestión de usuarios y de salas del sistema.

RU-F009	Esencial	Alta	El perfil administrador permitirá al usuario:
			• Gestión de usuarios del sistema.
			  ○ Creación de usuarios.
			  ○ Modificación y visualización de los datos de los usuarios:
			  ▪ Datos personales.
			  ▪ Contraseña.
			  ▪ Perfiles del usuario.
			  ○ Activación de nuevos usuarios del sistema.
			• Gestión de las salas del sistema.
			  ○ Modificación y visualización de los datos de la salas.
			  ▪ Nombre de la sala.
			  ▪ Privacidad de la sala.
			  ▪ Estado de la sala.
			  ▪ Tipo de participación.
			  ▪ Descripción de la sala.
			  ○ Inclusión/Exclusión de usuarios a las salas.
			  ○ Modificación de las funciones de los usuarios dentro de la sala.
			• Modificación y visualización de datos de

Identificador	Necesidad	Prioridad	Descripción
			usuario o Datos personales. o Contraseña de acceso. o Perfil del usuario. o Activo/Desactivo.
RU-F010	Esencial	Alta	El perfil diseñador permitirá al usuario: • Modificación y visualización de los datos del usuario: o Datos personales. o Contraseña de acceso. • Gestión de las salas del usuario. o Creación de salas. o Borrado de las salas creadas por el usuario. o Modificación y visualización de los datos de la salas creadas por el usuario. ▪ Nombre de la sala. ▪ Funciones de usuario en la sala. • Búsqueda de salas registradas en el sistema. • Participación en las salas del sistema.
RU-F011	Deseable	Media	Los datos personales de los usuarios serán los siguientes: • Nombre y apellidos. • Correo electrónico. • Nombre de usuario. • Password.
RU-F012	Esencial	Alta	El sistema deberá permitirá a los usuarios la creación de salas, unirse a salas ya creadas y ver las salas públicas que hay en ese momento en el sistema o las salas en las cuales está participando.
RU-F013	Esencial	Alta	Una sala dispondrá de las siguientes propiedades: • Una política de participación que determinara la forma en la cual los usuarios podrán usar la pizarra de la sala. • Una pizarra online síncrona. • Un chat multiusuario y síncrono.
RU-F014	Esencial	Alta	En la creación de la sala se solicitara al usuario: • Nombre de la sala. • Tipo de sala: Privada o pública. • Política de participación. Mediante la cual se define el modo de actuación de los distintos usuarios de la sala sobre la pizarra. • Descripción de la sala.
RU-F015	Esencial	Alta	El sistema tendrá las siguientes funciones para los usuarios de la aplicación con distintos permisos: • Propietario. • Moderador.

Identificador	Necesidad	Prioridad	Descripción
			• Colaborador. • Invitado.
RU-F016	Esencial	Alta	El usuario propietario tiene los siguientes permisos. • Modificar los datos de la sala. • Invitar usuarios a la sala. • Expulsar usuarios de la sala. • Modificar el rol de los otros usuarios. • Pintar en el lienzo. • Solicitar turno para usar el lienzo. • Usar el chat para comunicarse con el resto de usuarios. • Guardar el diseño. • Cargar diseños. • Salirse de la sala.
RU-F017	Esencial	Alta	El usuario moderador tiene los siguientes permisos. • Expulsar usuarios de la sala. • Invitar usuarios a la sala. • Modificar el rol de los otros usuarios. • Pintar en el lienzo. • Solicitar turno para usar el lienzo. • Usar el chat para comunicarse con el resto de usuarios. • Guardar el diseño. • Cargar diseños. • Salirse de la sala.
RU-F018		Alta	El usuario colaborador tiene los siguientes permisos. • Pintar en el lienzo. • Solicitar turno para usar el lienzo. • Usar el chat para comunicarse con el resto de usuarios. • Guardar el diseño. • Cargar diseños. • Salirse de la sala.
RU-F019	Esencial	Alta	El usuario invitado tiene los siguientes permisos. • Usar el chat para comunicarse con el resto de usuarios. • Salirse de la sala.
RU-F020	Esencial	Alta	La función del usuario se limita únicamente a la sala en la que se encuentre, pudiendo adoptar distintas funciones según en la sala en la que se encuentre.
RU-F021	Esencial	Alta	El creador de la sala tendrá siempre la función de propietario.
RU-F022	Esencial	Alta	El usuario propietario y los moderadores de la sala, podrán cambiar la función del resto de los usuarios en todo momento.
RU-F023	Deseable	Media	Cuando un usuario se una a una sala por primera

Identificador	Necesidad	Prioridad	Descripción
			vez se unirá con la función de colaborador.
RU-F024	Deseable	Media	Al unirse a una sala, se podrá acceder directamente a ella o solo añadirla a la lista de sus salas.
RU-F025	Deseable	Media	Cuando el propietario o los moderadores de una sala invita a otro usuario a unirse a la misma, se le enviara una solicitud de unión al usuario que se desea invitar, especificando la función que se desea que este tenga dentro de la sala y este podrá aceptar o declinar la invitación.
RU-F026	Deseable	Media	Los usuarios invitados por los propietarios de las salas, en caso de que acepten esta invitación, se unirán a la sala con la función especificada en la invitación
RU-F027	Deseable	Media	El contenido del lienzo se podrá exportar a imagen en cualquier momento.
RU-F028	Esencial	Media	El contenido del lienzo se podrá guardar en el sistema y se asociara al usuario que realice el guardado del mismo. Al guardar el diseño, se guardara como un conjunto de bytes en base de datos.
RU-F029	Deseable	Media	Se podrán cargar al lienzo imágenes externas, o diseños almacenados pertenecientes al usuario.
RU-F030	Esencial	Alta	Los usuarios administradores y moderadores podrán asociar un determinado diseño del lienzo a la sala. A partir de ese momento, al entrar en la sala será el diseño por defecto de la misma.
RU-F031	Esencial	Alta	Los usuarios administradores y moderadores podrán eliminar o sobrescribir un diseño de una sala.
RU-F032	Deseable	Media	Toda sala llevará un histórico del contenido del lienzo, pudiendo reproducir las distintas fases por las que ha pasado el diseño hasta el momento.

Tabla 16: Requisitos Funcionales

Requisitos no Funcionales

Identificador	Necesidad	Prioridad	Descripción
RU-NF001	Deseable	Alta	La interfaz web será adaptable para resoluciones de 1024*768 o superiores.
RU-NF002	Deseable	Media	El sistema deberá ser compatible con los navegadores: Firefox e Internet Explorer.
RU-NF003	Opcional	Baja	El sistema deberá ser compatible con los navegadores: Chrome y Opera.
RU-NF004	Deseable	Alta	El sistema deberá ser multilenguaje, y se podrá cambiar de lenguaje en cualquier momento.
RU-NF005	Deseable	Alta	Los idiomas que deberá soportar la aplicación son: Inglés y Español
RU-NF006	Deseable	Alta	El sistema deberá tener un perfil de administración para poder gestionar el sistema: a nivel de aplicación y de base de datos.
RU-NF007	Deseable	Alta	De forma adicional será necesario tener un usuario para la base de datos que actué como administrador para la gestión del sistema a este nivel.
RU-NF008	Deseable	Alta	Se definirán así dos usuarios para la base de datos del sistema: • Usuario administración, para la gestión interna de la base de datos, con todos los permisos. • Usuario para el sistema, para el uso de la base de datos desde la aplicación, con permisos de consultas, actualizaciones e inserciones en base de datos.
RU-NF009	Deseable	Alta	El sistema seguirá una arquitectura Cliente-Servidor, en el que la parte del cliente estará programada con Adobe Flex, y la parte del servidor con *JAVA*.
RU-NF011	Deseable	Alta	La comunicación entre servidor y cliente se realizara a través del software libre *BlazeDS*, sustitutivo del software propietario de Adobe, *GraniteDS*.
RU-NF011	Deseable	Alta	La base de datos usada en el sistema será *MySQL*.

Tabla 17: Requisitos no Funcionales

4.2.3. Especificación de requisitos

Para la especificación de los requisitos denominados de no usuario, nos centraremos en la definición de diversos casos de uso que engloben los requisitos definidos en el punto anterior.

De este modo también seguimos la especificación del modelo de desarrollo RUP, el cual, como se comentó con anterioridad, su pilar central son los casos de uso.

En la siguiente tabla se ve reflejado el inventario de los distintos casos de uso que se han tenido en cuenta.

Identificador	Nombre	Requisito/os
CU-001	*Acceso a la web de la aplicación*	RU-F001
CU-002	*Autenticación en el sistema*	RU-F002
CU-003	*Registro de nuevo usuario*	RU-F003 RU-F004 RU-F011
CU-004	*Acceso a la aplicación por parte de un nuevo usuario*	RU-F002 RU-F005
CU-005	*Visualización de usuarios del sistema*	RU-F006 RU-F007 RU-F009
CU-006	*Gestión de usuarios: Creación de nuevos usuarios*	RU-F004 RU-F006 RU-F007 RU-F009 RU-F011
CU-007	*Gestión de usuarios : Modificación de datos de usuario*	RU-F005 RU-F006 RU-F007 RU-F009
CU-008	*Visualización de salas del sistema*	RU-F006 RU-F007 RU-F009
CU-009	*Gestión de salas : Modificación de datos de sala*	RU-F006 RU-F007 RU-F009 RU-F013
CU-010	*Gestión de salas : Gestión de usuarios de sala*	RU-F006 RU-F007 RU-F009
CU-011	*Gestión de salas : Envío de invitaciones*	RU-F006 RU-F007 RU-F009
CU-012	*Gestión de salas : Expulsión de usuarios de una sala*	RU-F006 RU-F007 RU-F009

Identificador	Nombre	Requisito/os
CU-013	*Gestión de salas : Modificación de funciones de usuarios*	RU-F006 RU-F007 RU-F009
CU-014	*Visualización de datos de usuario*	RU-F006 RU-F007 RU-F010 RU-F011
CU-015	*Modificación de los datos personales de usuario*	RU-F006 RU-F007 RU-F010 RU-F011
CU-016	*Visualización de datos de usuario*	RU-F006 RU-F007 RU-F009 RU-F011
CU-017	*Modificación de los datos personales de usuario*	RU-F006 RU-F007 RU-F009 RU-F011
CU-018	*Visualización de salas personales del usuario*	RU-F006 RU-F007 RU-F010 RU-F012
CU-019	*Creación de una nueva sala*	RU-F006 RU-F007 RU-F010 RU-F012 RU-F013 RU-F014 RU-F021
CU-020	*Borrado de salas personales (I)*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F017 RU-F018 RU-F019 RU-F020
CU-021	*Borrado de salas personales (II)*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F020
CU-022	*Modificación de datos de una sala*	RU-F006 RU-F007 RU-F010

Identificador	Nombre	Requisito/os
		RU-F013 RU-F015 RU-F016 RU-F020
CU-023	*Gestión de usuarios de sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F020 RU-F022
CU-024	*Envío de invitaciones*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F020 RU-F022 RU-F025
CU-025	*Expulsión de usuarios de una sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F020 RU-F022
CU-026	*Modificación de funciones de usuarios*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F020 RU-F022
CU-027	*Acceso a una sala desde el menú de salas personales*	RU-F006 RU-F007 RU-F010
CU-028	*Visualización de salas públicas del sistema*	RU-F006 RU-F007 RU-F010 RU-F012
CU-029	*Apuntarse en una sala pública*	RU-F006 RU-F007 RU-F010

Identificador	Nombre	Requisito/os
		RU-F012 RU-F023 RU-F024
CU-030	*Acceso a una sala desde el menú de búsqueda de salas*	RU-F006 RU-F007 RU-F010 RU-F012 RU-F023 RU-F024
CU-031	*Pintar en el lienzo en una sala con participación "Uno a Uno"*	RU-F001 RU-F006 RU-F007 RU-F010 RU-F013 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020
CU-032	*Pintar en el lienzo en una sala con participación "Todos a la vez"*	RU-F001 RU-F006 RU-F007 RU-F010 RU-F013 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020
CU-033	*Solicitar el pincel en una sala con participación "Uno a Uno"*	RU-F001 RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020
CU-034	*Dejar el pincel en una sala con participación "Uno a Uno"*	RU-F001 RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020

Identificador	Nombre	Requisito/os
CU-035	*Guardar contenido del lienzo de una sala como imagen*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020 RU-F027
CU-036	*Cargar una imagen al contenido del lienzo de una sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020 RU-F029
CU-037	*Guardar el contenido del lienzo de una sala como diseño propio*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020 RU-F028
CU-038	*Cargar un diseño propio al contenido del lienzo de una sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F020 RU-F029
CU-039	*Comunicarse con otros usuarios mediante el chat de sala*	RU-F001 RU-F006 RU-F007 RU-F010 RU-F013 RU-F015 RU-F016 RU-F017 RU-F018 RU-F019 RU-F020

Identificador	Nombre	Requisito/os
CU-040	*Guardar un diseño para la sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F020 RU-F030
CU-041	*Eliminar un diseño de la sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F020 RU-F031
CU-042	*Visualización de acciones en el lienzo de la sala*	RU-F006 RU-F007 RU-F010 RU-F015 RU-F016 RU-F017 RU-F018 RU-F019 RU-F020
CU-043	*Visualización de invitaciones*	RU-F006 RU-F007 RU-F010 RU-F026 RU-F032

Tabla 18: Catalogo de Casos de Uso

A continuación se muestra de forma detallada cada uno de los casos de uso incluidos en el catalogo mostrado anteriormente:

Identificador	Nombre	Importancia	Actor
CU-001	*Acceso a la web de la aplicación*	Alta	Cualquier usuario
Descripción			
El usuario intenta acceder a la aplicación mediante un explorador web.			
Precondiciones			
Disponer en el PC de un explorador con el plug-in de Adobe Flash Player instalado.			
Flujo Normal de Eventos			
1- El usuario introduce la URL de la web en el explorador. 2- El explorador tiene el *plug-in* de Adobe Flash Player. 3- El explorador descarga los elementos necesarios de la web y la muestra al usuario. 4- El usuario se encuentra en la ventana de autenticación del sistema.			
Flujo Alternativo de Eventos			
2- El explorador no tiene el *plug-in* de Adobe Flash Player. 3- El explorador indica al usuario que para poder visualizar la web necesita descargarse e instalar el Adobe Flash Player. 4- El usuario descarga e instala el *plug-in* de Adobe Flash Player. 5- El usuario accede de nuevo a la aplicación. 6- El explorador descarga los elementos necesarios de la web y la muestra al usuario. 7- El usuario se encuentra en la ventana de autenticación del sistema.			

Tabla 19: Caso de Uso CU-001

Identificador	Nombre	Importancia	Actor
CU-002	*Autenticación en el sistema*	Alta	Usuario registrado
Descripción			
El usuario intenta acceder a la aplicación mediante el sistema de autenticación del sistema.			
Precondiciones			
CU-001			
Flujo Normal de Eventos			
1- El usuario introduce su nombre de usuario y su contraseña en el sistema. 2- El sistema verifica si las credenciales son correctas. 3- Credenciales correctas: El usuario accede al sistema.			
Flujo Alternativo de Eventos			
3- Credenciales incorrectas: El sistema deniega el acceso al usuario al sistema y mostrara al usuario "*El usuario con el que está intentando acceder es erróneo, por favor verifique los campos.*" 4- El usuario podrá volver a introducir sus usuarios para acceder al sistema.			

Tabla 20: Caso de Uso CU-002

Identificador	Nombre	Importancia	Actor
CU-003	*Registro de nuevo usuario*	Alta	Cualquier usuario
Descripción			
El usuario se registra dentro del sistema mediante el formulario de registro.			
Precondiciones			
CU-001			
Flujo Normal de Eventos			
1- El usuario accede al formulario de registro desde la página de autenticación del sistema. 2- El usuario rellena los campos que pide el formulario, y emite la solicitud de registro al sistema. Los datos que solicita el formulario son: • Nombre. • Apellidos. • Correo electrónico. • Nombre de usuario (username). • Contraseña (password). 3- Si los campos son correctos y el nombre de usuario no existe ya, el registro se completa con éxito.			
Flujo Alternativo de Eventos (I)			
3- Alguno de los campos es incorrecto. El formulario notifica al usuario cuales son los campos incorrectos para que este los verifique.			
Flujo Alternativo de Eventos (II)			
3- Los campos son correctos, pero el nombre de usuario ya está registrado en el sistema. 4- El sistema avisa al usuario de que el nombre de usuario ya está registrado.			

Tabla 21: Caso de Uso CU-003

Identificador	Nombre	Importancia	Actor
CU-004	*Acceso a la aplicación por parte de un nuevo usuario*	Alta	Usuario registrado
Descripción			
El usuario recién registrado intenta acceder a la aplicación.			
Precondiciones			
CU-001			
Flujo Normal de Eventos			
1- El usuario rellena el formulario de autenticación con su nombre de usuario y su contraseña. 2- El usuario emite la petición de acceso a la aplicación. 3- El sistema valida los datos. 4- El usuario es correcto y está activo: el sistema permite el acceso a la aplicación.			
Flujo Alternativo de Eventos (I)			
3- El usuario no está activado. 4- El sistema notifica al usuario que el usuario no es válido.			
Flujo Alternativo de Eventos (II)			
3- El usuario no ha introducido alguno de los datos correctamente. 4- El sistema notifica al usuario que el usuario no es válido.			

Tabla 22: Caso de Uso CU-004

Identificador	Nombre	Importancia	Actor
CU-005	*Visualización de usuarios del sistema*	Media	Usuario administrador
Descripción			
Consulta por parte de los usuarios administradores de los usuarios registrados en el sistema.			
Precondiciones			
CU-002			
Flujo Normal de Eventos			
1- El usuario accede al menú de administración. 2- El usuario accede al submenú de gestión de usuarios. 3- El usuario visualiza todos los usuarios registrados en ese momento en el sistema.			

Tabla 23: Caso de Uso CU-005

Identificador	Nombre	Importancia	Actor
CU-006	*Gestión de usuarios: Creación de nuevos usuarios*	Media	Usuario administrador
Descripción			
Creación por parte de usuarios administradores de nuevos usuarios.			
Precondiciones			
CU-002, CU-005			
Flujo Normal de Eventos			
1- Mediante el botón de la ventana de gestión de usuarios de "Añadir Usuario" el administrador podrá registrar a nuevos usuarios 2- A la hora de registrar un nuevo usuario el sistema presentara al usuario un formulario en el que solicitara lo siguientes datos: • Nombre. • Apellidos. • Correo electrónico. • Nombre de usuario (username). • Contraseña (password). 3- Si los campos son correctos y el nombre de usuario no existe ya, el registro se completa con éxito.			
Flujo Alternativo de Eventos (I)			
3- Alguno de los campos es incorrecto. El formulario notifica al usuario cuales son los campos incorrectos para que este los verifique. 4- El usuario puede modificarlos para intentar de nuevo el registro del usuario.			
Flujo Alternativo de Eventos (II)			
3- Los campos son correctos, pero el nombre de usuario ya está registrado en el sistema. 4- El sistema avisa al usuario de que el nombre de usuario ya está registrado. 5- El usuario puede modificar el nombre de usuario para intentar de nuevo el registro del usuario.			

Tabla 24: Caso de Uso CU-006

Identificador	Nombre	Importancia	Actor
CU-007	*Gestión de usuarios : Modificación de datos de usuario*	Media	Usuario administrador
Descripción			
Modificación por parte de usuarios administradores de los datos personales de un usuario.			
Precondiciones			
CU-002, CU-005			
Flujo Normal de Eventos			
1- El usuario podrá gestionar los datos de todos los usuarios del sistema haciendo click sobre el icono de datos de cada usuario. 2- El sistema presentara en un formulario los datos del usuario, pudiendo modificar entre ellos los siguientes: • Nombre. • Apellidos. • Correo electrónico. • Contraseña (password). • Modificar el perfil del usuario. • Activar/Desactivar usuarios. 3- El usuario actualiza los datos del usuario. 4- Si todos los datos son correctos el usuario se actualiza correctamente.			
Flujo Alternativo de Eventos (I)			
3- El usuario cancela actualiza los datos del usuario. 4- El sistema no actualiza los datos del usuario.			
Flujo Alternativo de Eventos (II)			
3- Alguno de los campos es incorrecto. 4- El sistema avisa al usuario de los campos incorrectos del formulario. 5- El usuario puede modificarlos para intentar de nuevo el registro del usuario.			

Tabla 25: Caso de Uso CU-007

Identificador	Nombre	Importancia	Actor
CU-008	*Visualización de salas del sistema*	Alta	Usuario administrador
Descripción			
Consulta por parte de los usuarios administradores de las salas registradas en el sistema.			
Precondiciones			
CU-002			
Flujo Normal de Eventos			
1- El usuario accede al menú de administración. 2- El usuario accede al submenú de gestión de salas. 3- El usuario visualiza todas las salas registradas en ese momento en el sistema.			

Tabla 26: Caso de Uso CU-008

Identificador	Nombre	Importancia	Actor
CU-009	*Gestión de salas : Modificación de datos de sala*	Alta	Usuario administrador
Descripción			
Modificación por parte de usuarios administradores de los datos de una sala.			
Precondiciones			
CU-002, CU-008			
Flujo Normal de Eventos			
1- El usuario podrá gestionar los datos de todas las salas del sistema haciendo click sobre el icono de datos de cada sala. 2- El sistema presentara en un formulario los datos de la sala, pudiendo modificar entre ellos los siguientes: • Nombre de la sala. • Privacidad de la sala. • Estado de la sala (abierta o cerrada). • Tipo de participación. • Descripción de la sala. 3- El usuario actualiza los datos de la sala. 4- El sistema actualiza los datos de la sala.			
Flujo Alternativo de Eventos (I)			
3- El usuario cancela actualiza los datos del usuario. 4- El sistema no actualiza los datos del usuario.			

Tabla 27: Caso de Uso CU-009

Identificador	Nombre	Importancia	Actor
CU-010	*Gestión de salas : Gestión de usuarios de sala*	Alta	Usuario administrador
Descripción			
Gestión de los usuarios participantes de una sala.			
Precondiciones			
CU-002, CU-008			
Flujo Normal de Eventos			
1- El usuario podrá gestionar los usuarios participantes de todas las salas del sistema haciendo click sobre el icono de usuarios de cada sala. 2- El sistema presentara en un formulario los usuarios que están participando en el sistema, y el resto de usuarios del sistema.			

Tabla 28: Caso de Uso CU-010

Identificador	Nombre	Importancia	Actor
CU-011	*Gestión de salas : Envío de invitaciones*	Alta	Usuario administrador
Descripción			
Inclusión de usuarios a una sala en la cual no se encuentren participando.			
Precondiciones			
CU-002, CU-008, CU-010			
Flujo Normal de Eventos			
1- El usuario puede enviar invitaciones de participación en una sala a aquellos usuarios que no estén participando ya en la sala. 2- El sistema solicitara al usuario que especifique con qué función desea incluir al usuario dentro de la sala. 3- El sistema enviara una invitación al usuario que se desea incluir en la sala notificándole que ha sido invitado. 4- El usuario invitado podrá aceptar o rechazar la invitación.			

Tabla 29: Caso de Uso CU-011

Identificador	Nombre	Importancia	Actor
CU-012	*Gestión de salas : Expulsión de usuarios de una sala*	Alta	Usuario administrador
Descripción			
Exclusión de usuarios participantes de una determinada sala.			
Precondiciones			
CU-002, CU-008, CU-010			
Flujo Normal de Eventos			
1- El usuario puede expulsar de una determinada sala a cualquier usuario no propietario que se encuentre participando en la sala.			

Tabla 30: Caso de Uso CU-012

Identificador	Nombre	Importancia	Actor
CU-013	*Gestión de salas : Modificación de funciones de usuarios*	Alta	Usuario administrador
Descripción			
Modificar la función de los usuarios participantes de una determinada sala.			
Precondiciones			
CU-002, CU-008, CU-010			
Flujo Normal de Eventos			
1- El usuario puede modificar la función que desempeña en una determinada sala cualquier usuario que se encuentre participando en la sala. 2- A la hora de cambiar la función el sistema mostrara la función actual del usuario y las distintas opciones de funciones dentro de la sala. 3- El usuario ha de actualizar la sala para guardar los cambios 4- El sistema registra los cambios realizados por el usuario.			
Flujo Alternativo de Eventos			
3- El usuario cancela los cambios. 4- El sistema no registra los cambios realizados por el usuario.			

Tabla 31: Caso de Uso CU-013

Identificador	Nombre	Importancia	Actor
CU-014	*Visualización de datos de usuario*	Media	Usuario diseñador
Descripción			
Permisos del usuario diseñador en referencia a la visualización de sus datos de usuario.			
Precondiciones			
CU-002			
Flujo Normal de Eventos			
1- El usuario accede al menú de usuarios. 2- Desde el submenú de mis datos, el usuario puede visualizar sus datos personales. 3- El sistema presentara un formulario al usuario con todos sus datos: • Nombre. • Apellidos. • Correo electrónico. • Nombre de usuario. • Contraseña (password).			

Tabla 32: Caso de Uso CU-014

Identificador	Nombre	Importancia	Actor
CU-015	*Modificación de los datos personales de usuario*	Media	Usuario diseñador
Descripción			
Permisos del usuario diseñador en referencia a modificación de sus datos personales.			
Precondiciones			
CU-002			
Flujo Normal de Eventos			
1- El usuario accede al menú de usuarios. 2- Desde el submenú de mis datos, el usuario puede modificar sus datos personales. 3- El sistema presentara un formulario al usuario con todos sus datos, pudiendo modificar de estos los siguientes: • Nombre. • Apellidos. • Correo electrónico. • Contraseña (password).			

Tabla 33: Caso de Uso CU-015

Identificador	Nombre	Importancia	Actor
CU-016	*Visualización de datos de usuario*	Media	Usuario diseñador - administrador
Descripción			
Permisos del usuario diseñador-administrador en referencia a la visualización de sus datos de usuario.			
Precondiciones			
CU-002, CU-014			
Flujo Normal de Eventos			
1- El usuario administrador también puede visualizar los siguientes datos: • Perfiles del usuario. • Activo/Desactivo.			

Tabla 34: Caso de Uso CU-016

Identificador	Nombre	Importancia	Actor
CU-017	*Modificación de los datos personales de usuario*	Media	Usuario diseñador - administrador
Descripción			
Permisos del usuario administrador en referencia a la modificación de sus datos de usuario.			
Precondiciones			
CU-002, CU-015			
Flujo Normal de Eventos			
1- El usuario administrador también puede modificar los siguientes datos: • Perfiles del usuario. • Activo/Desactivo.			

Tabla 35: Caso de Uso CU-017

Identificador	Nombre	Importancia	Actor
CU-018	*Visualización de salas personales del usuario*	Alta	Usuario diseñador
Descripción			
Visualización de las salas en las cuales se encuentra participando el usuario en el sistema.			
Precondiciones			
CU-002			
Flujo Normal de Eventos			
1- El usuario accede al menú de salas. 2- El usuario accede al submenú de salas personales. 3- El usuario visualiza todas las salas en las cuales se encuentra participando dentro del sistema.			

Tabla 36: Caso de Uso CU-018

Identificador	Nombre	Importancia	Actor
CU-019	*Creación de una nueva sala*	Alta	Usuario diseñador
Descripción			
Creación de una nueva sala.			
Precondiciones			
CU-002, CU-018			
Flujo Normal de Eventos			
1- El usuario accede al formulario de creación de una nueva sala mediante el botón de "Crear Sala". 2- El usuario rellena los campos que pide el formulario, y emite la solicitud de creación al sistema. • Nombre de la sala. • Tipo de sala: Privada o Pública. • Política de participación: Forma mediante la cual se define el modo de actuación de los distintos usuarios de la sala sobre la pizarra. • Descripción de la sala. 3- Si los campos son correctos y el usuario no tenía una sala con el mismo nombre, la creación se completa con éxito. 4- El usuario creador obtiene directamente la función de propietario de la sala creada.			
Flujo Alternativo de Eventos (I)			
3- Alguno de los campos es incorrecto. El formulario notifica al usuario cuales son los campos incorrectos para que este los verifique. 4- El usuario revisa los campos comentados y completa el registro de la nueva sala. 5- El usuario creador obtiene directamente la función de propietario de la sala creada.			
Flujo Alternativo de Eventos (II)			
3- Los campos son correctos, pero el usuario ya tiene una sala asociada con el mismo nombre. 4- El sistema avisa al usuario de que ya existe una sala con el mismo nombre asociada al usuario.			

Tabla 37: Caso de Uso CU-019

Identificador	Nombre	Importancia	Actor
CU-020	*Borrado de salas personales (I)*	Alta	Usuario diseñador Función: • Moderador. • Colaborador. • Invitado.
Descripción			
Borrado de salas personales de un usuario diseñador.			
Precondiciones			
CU-002, CU-018			
Flujo Normal de Eventos			
1- El usuario puede dejar de participar en las salas en las que está participando pulsando el botón de borrar sala de cualquier de las salas personales del usuario. 2- El sistema advertirá al usuario si realmente desea dejar de participar en la sala. 3- El usuario acepta y la sala se borra de sus salas personales.			
Flujo Alternativo de Eventos (I)			
3- El usuario cancela y la sala se mantiene en sus salas personales			

Tabla 38: Caso de Uso CU-020

Identificador	Nombre	Importancia	Actor
CU-021	*Borrado de salas personales (II)*	Alta	Usuario diseñador Función: • Propietario.
Descripción			
Un usuario propietario deja de participar en una sala.			
Precondiciones			
CU-002, CU-018			
Flujo Normal de Eventos			
1- El usuario puede dejar de participar en las salas en las que está participando pulsando el botón de borrar sala de cualquier de las salas personales del usuario. 2- El sistema muestra una advertencia de que si se borra la sala se borrara la sala y todas las relaciones de esta sala con el resto de usuarios. 3- El usuario acepta: se borra la sala y esta desaparece de las salas personales de todos los usuarios.			
Flujo Normal de Eventos			
3- El usuario cancela y la sala se mantiene en sus salas personales			

Tabla 39: Caso de Uso CU-021

Identificador	Nombre	Importancia	Actor
CU-022	*Modificación de datos de una sala*	Alta	Usuario diseñador Función • Propietario.
Descripción			
Modificación de los datos de salas personales creadas por un usuario diseñador.			
Precondiciones			
CU-002, CU-018			
Flujo Normal de Eventos			
1- El usuario podrá modificar los siguiente datos de las salas en las cuales está participando como propietario: • Nombre de la sala. • Privacidad de la sala. • Estado de la sala (abierta o cerrada). • Tipo de participación. • Descripción de la sala.			

Tabla 40: Caso de Uso CU-022

Identificador	Nombre	Importancia	Actor
CU-023	*Gestión de usuarios de sala*	Alta	Usuario diseñador Función • Propietario. • Moderador.
Descripción			
Gestión de los usuarios participantes de una sala.			
Precondiciones			
CU-002, CU-018			
Flujo Normal de Eventos			
1- El usuario podrá gestionar los usuarios participantes de todas las salas del sistema haciendo click sobre el icono de usuarios de cada sala. 2- El sistema presentara en un formulario los usuarios que están participando en el sistema, y el resto de usuarios del sistema.			

Tabla 41: Caso de Uso CU-023

Identificador	Nombre	Importancia	Actor
CU-024	*Envío de invitaciones*	Alta	Usuario diseñador Función • Propietario. • Moderador.
Descripción			
Inclusión de usuarios a una sala en la cual no se encuentren participando.			
Precondiciones			
CU-002, CU-018, CU-023			
Flujo Normal de Eventos			
1- El usuario puede enviar invitaciones de participación en una sala a aquellos usuarios que no estén participando ya en la sala. 2- El sistema solicitara al usuario que especifique con qué función desea incluir al usuario dentro de la sala. 3- El sistema enviara una invitación al usuario que se desea incluir en la sala notificándole que ha sido invitado. 4- El usuario invitado podrá aceptar o rechazar la invitación.			

Tabla 42: Caso de Uso CU-024

Identificador	Nombre	Importancia	Actor
CU-025	*Expulsión de usuarios de una sala*	Alta	Usuario diseñador Función • Propietario. • Moderador.
Descripción			
Exclusión de usuarios participantes de una determinada sala.			
Precondiciones			
CU-002, CU-018, CU-023			
Flujo Normal de Eventos			
1- El usuario puede expulsar de una determinada sala a cualquier usuario no propietario que se encuentre participando en la sala.			

Tabla 43: Caso de Uso CU-025

Identificador	Nombre	Importancia	Actor
CU-026	*Modificación de funciones de usuarios*	Alta	Usuario diseñador Función • Propietario. • Moderador.
Descripción			
Modificar la función de los usuarios participantes de una determinada sala.			
Precondiciones			
CU-002, CU-018, CU-023			
Flujo Normal de Eventos			
1- El usuario puede modificar la función que desempeña en una determinada sala cualquier usuario que se encuentre participando en la sala. 2- A la hora de cambiar la función el sistema mostrara la función actual del usuario y las distintas opciones de funciones dentro de la sala. 3- El usuario ha de actualizar la sala para guardar los cambios 4- El sistema registra los cambios realizados por el usuario.			
Flujo Alternativo de Eventos			
3- El usuario cancela los cambios. 4- El sistema no registra los cambios realizados por el usuario.			

Tabla 44: Caso de Uso CU-026

Identificador	Nombre	Importancia	Actor
CU-027	*Acceso a una sala desde el menú de salas personales*	Alta	Usuario diseñador
Descripción			
Acceso a una sala en la cual está participando el usuario.			
Precondiciones			
CU-002, CU-018			
Flujo Normal de Eventos			
1- El usuario puede acceder a cualquiera de las salas en las que se encuentra participando desde el menú de salas personales al hacer doble click sobre la sala. 2- Al acceder el sistema cargara en una nueva pestaña la sala a la cual ha accedido el usuario.			

Tabla 45: Caso de Uso CU-027

Identificador	Nombre	Importancia	Actor
CU-028	*Visualización de salas públicas del sistema*	Alta	Usuario diseñador
Descripción			
Visualización de las salas públicas del sistema.			
Precondiciones			
CU-002			
Flujo Normal de Eventos			
1- El usuario accede al menú de salas. 2- El usuario accede al submenú de búsqueda de salas. 3- El usuario visualiza todas las salas públicas registradas en el sistema.			

Tabla 46: Caso de Uso CU-028

Identificador	Nombre	Importancia	Actor
CU-029	*Apuntarse en una sala pública*	Alta	Usuario diseñador
Descripción			
Apuntarse para empezar a participar en una sala pública del sistema.			
Precondiciones			
CU-002, CU-028			
Flujo Normal de Eventos			
1- El usuario puede apuntarse a cualquier sala pública del sistema desde la ventana de búsqueda de salas. 2- Una vez que el usuario se apunta a una sala esta pasa a formar parte de sus salas personales. 3- El usuario adquiere la función de colaborador en la sala en la que se apunta.			

Tabla 47: Caso de Uso CU-029

Identificador	Nombre	Importancia	Actor
CU-030	*Acceso a una sala desde el menú de búsqueda de salas*	Alta	Usuario diseñador
Descripción			
Acceso a una sala publica del sistema en la cual el usuario no está participando.			
Precondiciones			
CU-002, CU-028			
Flujo Normal de Eventos			
1- El usuario puede acceder a cualquier sala pública del sistema desde la ventana de búsqueda de salas. 2- Al acceder el sistema cargara en una nueva pestaña la sala a la cual ha accedido el usuario. 3- Una vez que el usuario se accede a la sala esta pasa a formar parte de sus salas personales. 4- El usuario adquiere la función de colaborador en la sala en la que se apunta.			

Tabla 48: Caso de Uso CU-030

Identificador	Nombre	Importancia	Actor
CU-031	*Pintar en el lienzo en una sala con participación "Uno a Uno"*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Pintar en una sala cuya política de participación es "Uno a Uno".			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario intenta pintar en el lienzo. 2- El usuario tiene el pincel. 3- El lienzo registra correctamente los trazos realizados por el usuario. 4- Las acciones realizadas por el usuario se replican correctamente en el lienzo del resto de usuarios conectados en la sala.			
Flujo Alternativo de Eventos			
2- El usuario no tiene el pincel. 3- El usuario no puede dibujar en el lienzo de la sala. 4- Las acciones realizadas por el usuario propietario del pincel se replican correctamente en el lienzo del usuario.			

Tabla 49: Caso de Uso CU-031

Identificador	Nombre	Importancia	Actor
CU-032	*Pintar en el lienzo en una sala con participación "Todos a la vez"*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Pintar en una sala cuya política de participación es "Todos a la vez".			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- Tras entrar en la sala el usuario puede pintar directamente en el lienzo de la sala. 2- Las acciones realizadas por el usuario se replican correctamente en el lienzo del resto de usuarios conectados en la sala.			

Tabla 50: Caso de Uso CU-032

Identificador	Nombre	Importancia	Actor
CU-033	*Solicitar el pincel en una sala con participación "Uno a Uno"*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Solicitar el pincel para realizar acciones sobre el lienzo de la sala.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede solicitar el pincel en cualquier momento siempre y cuando no tenga ya el pincel. 2- La petición del pincel se notificara al resto de usuarios de la sala. 3- La petición se encolara y gestionara en el servidor. 4- Una vez que el pincel se libere este pasara al siguiente usuario que toque según la política de rotación implementada (por defecto FIFO). 5- Cuando el usuario recibe el pincel ya puede realizar acciones sobre el lienzo de la sala.			

Tabla 51: Caso de Uso CU-033

Identificador	Nombre	Importancia	Actor
CU-034	*Dejar el pincel en una sala con participación "Uno a Uno"*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Dejar el pincel para que este rote entre los usuarios que han solicitado el pincel previamente.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede dejar el pincel en cualquier momento siempre y cuando lo tenga. 2- Al dejar el pincel el sistema obtendrá el siguiente propietario que solicito el pincel y se lo pasara a dicho usuario. 3- A partir de este momento siguiente propietario del pincel podrá realizar acciones sobre el lienzo. 4- El usuario que dejo el pincel puede solicitar de nuevo el pincel.			
Flujo Alternativo de Eventos			
3- En caso de no existir un siguiente propietario, el siguiente usuario que realice el pincel obtendrá el pincel de forma automática. 4- El usuario que dejo el pincel puede solicitar de nuevo el pincel.			

Tabla 52: Caso de Uso CU-034

Identificador	Nombre	Importancia	Actor
CU-035	*Guardar contenido del lienzo de una sala como imagen*	Media	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Guardar el contenido del lienzo de una sala como imagen a fichero local.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede exportar a una imagen en fichero local en contenido del lienzo en cualquier momento. 2- El sistema solicitara al usuario indicar la ruta en la cual desea guardar la nueva imagen.			

Tabla 53: Caso de Uso CU-035

Identificador	Nombre	Importancia	Actor
CU-036	*Cargar una imagen al contenido del lienzo de una sala*	Media	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Cargar una imagen de un fichero local al contenido del lienzo de la sala.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede realizar acciones sobre el lienzo. 2- El usuario puede cargar una imagen de un fichero local al contenido del lienzo de la sala. 3- El sistema pide al usuario que indique la ruta en la cual se encuentra la imagen que desea importar al contenido del lienzo.			

Tabla 54: Caso de Uso CU-036

Identificador	Nombre	Importancia	Actor
CU-037	*Guardar el contenido del lienzo de una sala como diseño propio*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Guardar el contenido del lienzo de una sala como un diseño propio del usuario.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede exportar el contenido del lienzo a un diseño propio en cualquier momento. 2- El sistema solicitara al usuario indicar el nombre que le quiere dar al nuevo diseño.			

Tabla 55: Caso de Uso CU-037

Identificador	Nombre	Importancia	Actor
CU-038	*Cargar un diseño propio al contenido del lienzo de una sala*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador.
Descripción			
Cargar un diseño propio al contenido del lienzo de la sala.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede realizar acciones sobre el lienzo. 2- El usuario puede cargar un diseño propio al contenido del lienzo. 3- El sistema mostrara en una ventana los distintos lienzos que el usuario tiene almacenados. 4- Al seleccionar un diseño y cargarlo, el contenido será automáticamente el contenido del diseño almacenado.			

Tabla 56: Caso de Uso CU-038

Identificador	Nombre	Importancia	Actor
CU-039	*Comunicarse con otros usuarios mediante el chat de sala*	Alta	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador. • Invitado.
Descripción			
Cargar un diseño propio al contenido del lienzo de la sala.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario puede usar el chat para comunicarse con otros usuarios de la sala. 2- El usuario escribe algo en el chat de la sala y el resto de usuarios y el mismo ven lo escrito por el usuario. 3- El usuario recibe lo que escriben el resto de usuarios de la sala.			

Tabla 57: Caso de Uso CU-039

Identificador	Nombre	Importancia	Actor
CU-040	*Guardar un diseño para la sala*	Alta	Usuario diseñador Función: • Propietario. • Moderador.
Descripción			
Guardar un diseño representado en un lienzo en base de datos asociándoselo a la sala.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario pulsa el botón de "Asociar diseño a la sala". 2- El sistema asocia a la sala el nuevo diseño. En caso de que otro diseño estuviese asociado, se sobrescribe. 3- A partir de este momento el lienzo de la sala será el diseño guardado.			

Tabla 58: Caso de Uso CU-040

Identificador	Nombre	Importancia	Actor
CU-041	*Eliminar un diseño de la sala*	Alta	Usuario diseñador Función: • Propietario. • Moderador.
Descripción			
Eliminar el diseño asociado a una sala.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario pulsa el botón de "Eliminar diseño de la sala". 2- El sistema borra la asociación que existe entre la sala y el diseño guardado anteriormente. 3- A partir de este momento el lienzo de la sala será vacio.			

Tabla 59: Caso de Uso CU-041

Identificador	Nombre	Importancia	Actor
CU-042	*Visualización de acciones en el lienzo de la sala*	Media	Usuario diseñador Función: • Propietario. • Moderador. • Colaborador. • Invitado.
Descripción			
Ver las acciones realizadas dentro del lienzo de la sala a lo largo del tiempo que el usuario lleva dentro de la misma.			
Precondiciones			
CU-002, CU-027 ó CU-030			
Flujo Normal de Eventos			
1- El usuario ve en todo momento las distintas acciones realizadas en el lienzo 2- Usando el botón deshacer vera los pasos previos. 3- Usando el botón rehacer vera los pasos posteriores.			

Tabla 60: Caso de Uso CU-042

Identificador	Nombre	Importancia	Actor
CU-043	*Visualización de invitaciones*	Media	Usuario diseñador
Descripción			
Un usuario puede ver en todo momento las distintas invitaciones a salas que tiene.			
Precondiciones			
El usuario ha accedido al sistema.			
Flujo Normal de Eventos			
1- El usuario accede a su buzón mediante el botón de notificación de la parte superior de la pantalla. 2- En esa ventana puede observar las distintas invitaciones que tiene el usuario. 3- Desde esa ventana puede aceptar o declinar la invitación. 4- El usuario acepta la invitación, se crea la relación sala-usuario con la función especificada en la invitación y ya puede ser visible desde el menú de salas personales. 5- El usuario declina la invitación, esta se borra y no tiene más repercusión sobre el sistema.			

Tabla 61: Caso de Uso CU-043

4.2.4. Diseño

El diseño es el segundo paso en la fase de desarrollo de cualquier producto de ingeniería, cuyo objetivo es anticipar y guiar el proceso de producción, produciendo modelos o representaciones de aquellas entidades que posteriormente se van a construir.

En este caso el diseño de la aplicación se ha llevado mediante la realización de distintos modelos de sistemas según se iban realizando las distintas iteraciones en el proceso que describe la metodología RUP. A lo largo de los siguientes puntos se describirán cada uno de los modelos realizados a lo largo del ciclo de vida del desarrollo, y se analizara la arquitectura básica del sistema.

4.2.5. Diseño de sistema

El sistema que se va a desarrollar se plantea como una aplicación, con lo que ha de ser contemplada desde dos perspectivas distintas: la parte cliente y la parte servidora.

De forma esquemática, el sistema seguirá la siguiente arquitectura:

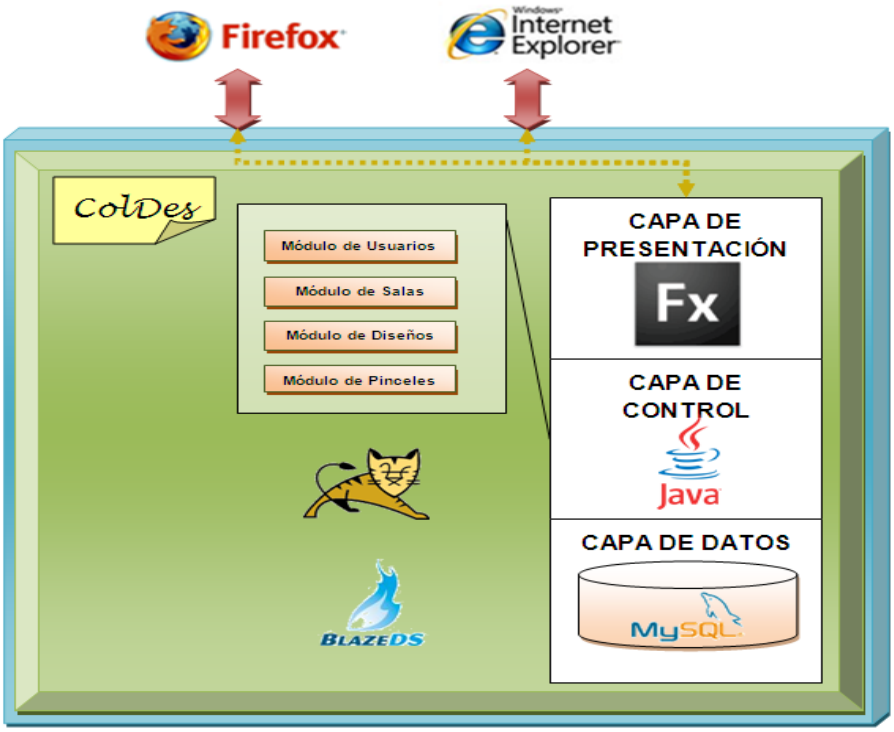


Figura 18: Arquitectura de ColDes

El sistema planteado es una aplicación web que, por tanto, será accesible desde cualquier navegador web estándar. Como se indico anteriormente en los requisitos, se garantizará el correcto funcionamiento de la aplicación bajo los dos navegadores más comunes: Mozilla Firefox e Internet Explorer.

La aplicación web ColDes, residirá en un servidor de aplicaciones Apache Tomcat, servidor libre sujeto a la licencia "*Apache Software License*" (*http://www.apache.org/licenses/*). Este servidor de aplicaciones será el proporcionado por el sistema de intercambio de datos OpenSource BlazeDS, que permite el intercambio de información de forma eficiente y segura entre un lado servidor basado en Java y un lado cliente basado en Flex y ActionScript.

El sistema empleará una arquitectura de tres capas, descritas a continuación:

- *Capa 1, de datos*. Esta capa cubre todo lo relacionado con el almacenamiento y la persistencia de la información. Para ello se empleará el Sistema de Gestión de Bases de Datos MySQL Server Versión 5.1. Se trata de un servidor de bases de datos gratuito y bastante maduro recientemente adquirido por la empresa Oracle, líder mundial en sistemas de gestión de bases de datos. La comunicación entre esta capa y la capa 2, de control, se realizará mediante JDBC.
- *Capa 2, de control*. Esta capa cubre la lógica de la aplicación, los procesos internos y la gestión de la información. Sirve asimismo de intermediaria entre la capa 1, de datos, y la capa 3, de presentación. Está desarrollada plenamente en el lenguaje Java, presente en la mayoría de las aplicaciones web existentes en el mercado, perteneciente a la importante empresa Sun Microsystems. La comunicación entre esta capa y la capa 3, de presentación, se realizará mediante la interfaz BlazeDS, versión OpenSource de la interfaz comercial LifeCycle® Enterprise Suite, de la empresa Adobe.
- *Capa 3, de presentación*. Esta capa cubre toda la interfaz gráfica de usuario: formularios, cuadros de texto, listas desplegables, tablas, ventanas… Toda ella se desarrollará en Adobe Flex, dada su elevada usabilidad y sus resultados espectaculares.

Parte servidora

La parte servidora estará basada en servlets Java basados en la plataforma J2EE Versión 5.

Los servlets residirán en un contenedor de servlets basado en Apache Tomcat Versión 6. Sin embargo, no se empleará la versión original de Apache Foundation, sino una alternativa Open Source proporcionada por Adobe Systems, denominada BlazeDS que incluye:

- Contenedor de servlets Java basado en Apache Tomcat 6, pero optimizado para el funcionamiento con Adobe Flex (ver parte cliente).
- Interfaces para conectar la parte servidora, desarrollada en Java, con la parte cliente, desarrollada en Flex (ver parte cliente).

Parte cliente

Por tratarse de una aplicación web, la parte cliente será accesible mediante cualquier navegador compatible con Adobe Flash V. 9 ó superior. Se garantizará el correcto funcionamiento de la aplicación bajo los navegadores más habituales, Internet Explorer y Mozilla Firefox,

La interfaz de usuario está desarrollada mediante Adobe Flex, que permite al mismo tiempo un desarrollo ágil de interfaces de usuario y la obtención de interfaces muy ricas, que proporcionan una agradable experiencia de usuario y hacen que la aplicación tenga una elevada usabilidad.

Además, dado que el usuario únicamente ve en su navegador una "película" flash, no existen diferencias en el aspecto de los elementos entre unos navegadores y otros.

Relaciones con Otros Sistemas

Se establece como requisito para el funcionamiento de la aplicación en la parte cliente que los navegadores empleados tengan instalado el *plug-in* gratuito de Adobe Flash Player, que permite a los navegadores reproducir películas flash (formato SWF), formato en el que se presenta la aplicación a los usuarios.

Este plug-in puede descargarse de forma gratuita e instalarse con facilidad. Para obtenerlo y consultar instrucciones detalladas de instalación, puede consultarse la página oficial de Adobe: http://www.adobe.es.

4.2.6. Diseño detallado

A lo largo de este capítulo de la memoria veremos de forma detallada el diseño de las distintas capas que presenta el sistema (datos, control y presentación):

Figura 19: Capas del sistema

El sistema que se ha desarrollado sigue un claro modelo – vista – controlador. El Modelo Vista Controlador (*MVC*) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. En nuestra aplicación, la vista seria la parte *Flex* del sistema, el modelo seria la base de datos alojada en la *MySQL*, y el controlador contendría toda la lógica de control del sistema, desarrollado en el lenguaje de programación *JAVA*.

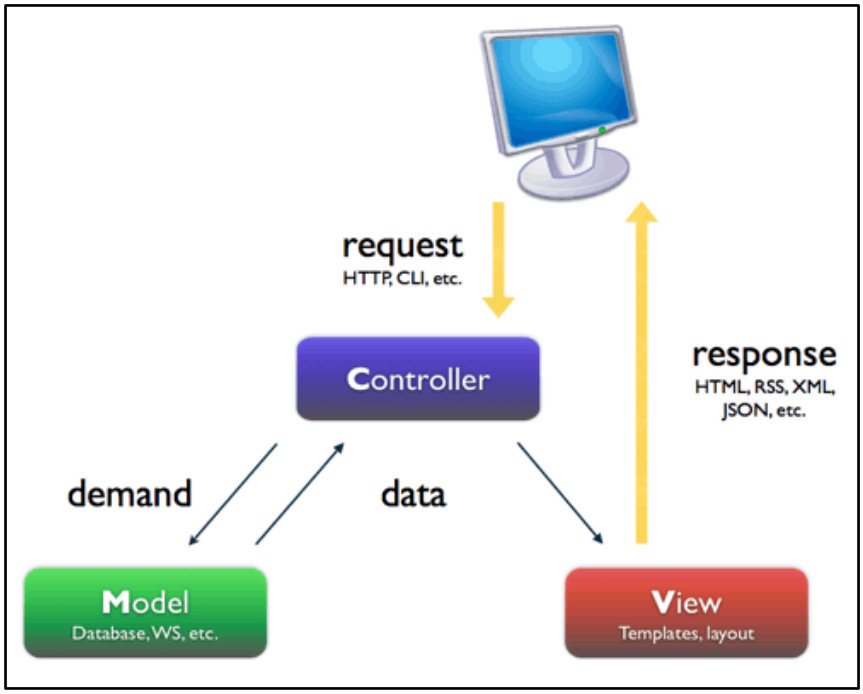


Figura 20: Modelo Vista Controlador (MVC)

Capa de datos

En esta capa presentaremos el modelo de persistencia seleccionado para dar soporte a la aplicación completa. Esta capa cubre todo lo relacionado con el almacenamiento y el almacenamiento de la información.

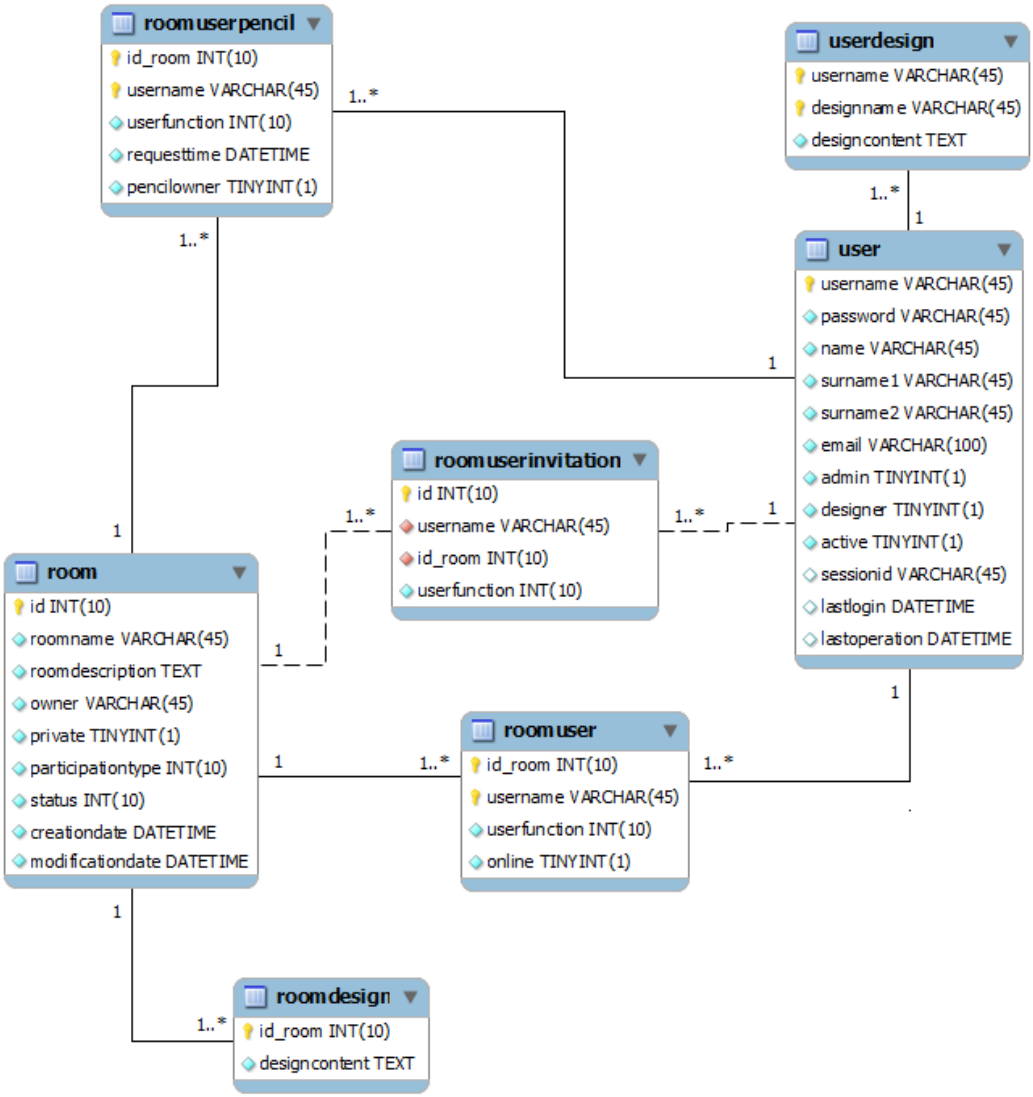
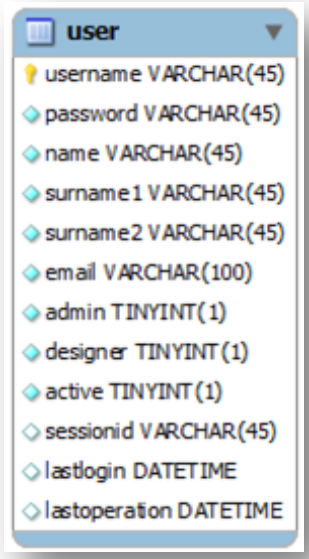


Figura 21: Modelo de E/R (UML)

A continuación comentaremos cada una de las tablas presentes en el sistema, intentan explicar lo más detalladamente posible su función y los distintos campos que almacenan:

- *Tabla **User**:*

En esta tabla se almacenaran todos los datos referentes a los usuarios de la aplicación. La clave primaria de esta tabla será el *username*, que identificara unívocamente al usuario. Junto con esta tabla se creara un índice, cuya clave será el campo *username*, para facilitar las búsquedas de usuarios en el sistema.

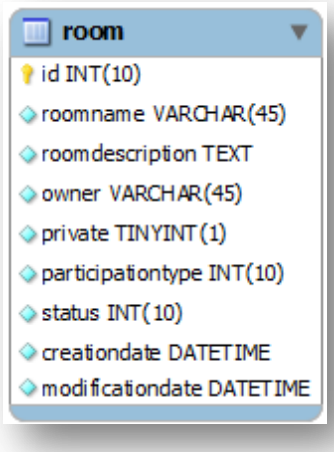


USER		
Campo	Tipo	Descripción
username	*varchar(45)*	Nombre mediante el cual se identificara al usuario dentro de la aplicación. Este campo actúa como clave primaria de la tabla.
password	*varchar(45)*	Contraseña que con la cual el usuario podrá entrar en la aplicación. Lo que se almacenara en base de datos será un hash de la contraseña del usuario.
name	*varchar(45)*	Nombre real del usuario.
surname1	*varchar(45)*	Primer apellido del usuario.
surname2	*varchar(45)*	Segundo apellido del usuario.
email	*varchar(100)*	Correo electrónico del usuario.
admin	*tynyint(1)*	Campo booleano que indica si el usuario es administrador
designer	*tynyint(1)*	Campo booleano que indica si el usuario es diseñador.
active	*tynyint(1)*	Campo booleano que indica si el usuario esta o no activo.
sessionid	*varchar(45)*	Valor que tiene la sesión actual del usuario en caso de encontrase autenticado en el sistema. De no ser así este campo tendrá valor *null*.
lastlogin	*datetime*	Fecha que indica la última vez que el usuario estuvo autenticado en el sistema.
lastoperation	*datetime*	Fecha que indica la última operación del usuario dentro del sistema

Tabla 62: Tabla User

- *Tabla **Room**:*

Esta tabla registrara los datos de todas las salas creadas en el sistema. Para esta tabla se usara un *id* numérico auto-incrementable como clave. Así mismo se generaran dos índices distintos para agilizar la búsqueda de estas dentro del sistema, uno por el *id* de la sala y otro por el *username* del propietario de la sala (*owner*).

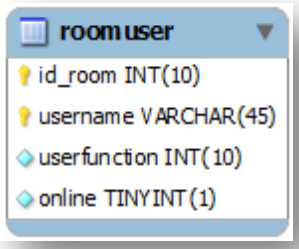


ROOM		
Campo	Tipo	Descripción
id	*int(10)*	Valor numérico auto-incrementable que sirve de clave para esta tabla.
roomname	*varchar(45)*	Nombre de la sala.
roomdescription	*text*	Descripción de la sala
owner	*varchar(45)*	Nombre de usuario del propietario de la sala. Este campo es una clave ajena (Foreign Key) que hace referencia al campo *username* de la tabla **User**.
prívate	*tynyint(1)*	Campo booleano que indica si la sala es o no privada.
participationtype	*int(10)*	Indica el tipo de participación que soporta la sala Los posibles valores son: • 0→ Uno a uno. • 1→ Todos a la vez.
status	*int(10)*	Indica el estado de la sala (abierta o cerrada).
creationdate	*datetime*	Fecha que indica el momento de creación de la sala.
modificationdate	*datetime*	Fecha que indica la última modificación realizada dentro de la sala.

Tabla 63: Tabla Room

- *Tabla **RoomUser**:*

Esta tabla registrará las asociaciones entre las distintas salas y los usuarios del sistema. De esta tabla podremos obtener tanto los usuarios pertenecientes a una sala y su función dentro de la misma, como las salas asociadas a un usuario y las funciones que desempeña en ellas. De esta forma, la clave primaria de esta tabla será el *id* de la sala y el *username* del usuario. Esta tabla tendrá varios índices distintos, uno por el id de sala y otro por el *username* del usuario.

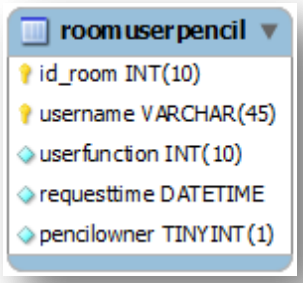


ROOMUSER		
Campo	Tipo	Descripción
id_room	*int(10)*	Id de la sala a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *id* de la tabla **Room**.
username	*varchar(45)*	Nombre de usuario a la cual hace referencia esta relación. Es una clave ajena que hace referencia al campo *username* de la tabla **User**.
userfunction	*int(10)*	Función que desempeña el usuario dentro de la sala. Los posibles valores de este campo pueden ser: • 0 → Propietario. • 1 → Moderador. • 2 → Colaborador. • 3 → Invitado.
online	*tynyint(1)*	*Indica si el usuario esta dentro de la sala o no.*

Tabla 64: Tabla RoomUser

- *Tabla **RoomUserPencil**:*

Esta tabla registrará las distintas peticiones de uso del pincel dentor de cada una de las salas. La clave primaria de esta tabla sera la compuesta por el *id* de la sala y el *username* del usuario que realiza la petición. Esta tabla tendrá varios incides distintos, uno por el id de sala y otro por el *username* del usuario.

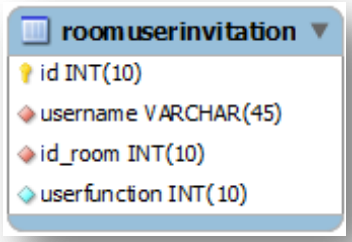


ROOMUSERPENCIL		
Campo	Tipo	Descripción
id_room	*int(10)*	Id de la sala a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *id* de la tabla **Room**.
username	*varchar(45)*	Nombre de usuario a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *username* de la tabla **User**.
userfunction	*int(10)*	Función que desempeña el usuario que realiza la petición del pincel dentro de la sala. Los posibles valores de este campo pueden ser: • 0 → Propietario. • 1 → Moderador. • 2 → Colaborador. • 3 → Invitado.
requesttime	*datetime*	Momento en el que se realiza la petición del pincel en la sala.
pencilowner	*tynyint(1)*	Indica si el usuario de la petición es el actual propietario del pincel.

Tabla 65: Tabla RoomUserPencil

- *Tabla **RoomUserInvitation**:*

Esta tabla registrará los envios de invitaciones a salas entre los distintos usuarios del sistema.

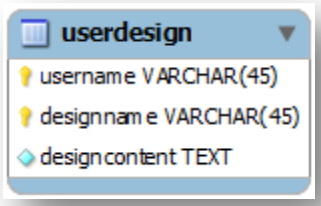


ROOMUSERINVITATION		
Campo	Tipo	Descripción
id	*int(10)*	Valor numérico auto-incrementable que sirve de clave para esta tabla.
username	*varchar(45)*	Nombre de usuario a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *username* de la tabla **User**.
id_room	*int(10)*	Id de la sala a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *id* de la tabla **Room**.
userfunction	*int(10)*	Función que desempeña el usuario que realiza la petición del pincel dentro de la sala. Los posibles valores de este campo pueden ser: • 0 → Propietario. • 1 → Moderador. • 2→ Colaborador. • 3 → Invitado.

Tabla 66: Tabla RoomUserInvitation

- *Tabla **UserDesign**:*

En esta tabla se almacenaran los diseños que los usuarios quieran guardar en la base de datos. Para evitar dos diseños nombrados de la misma forma por parte de un mismo usuario, la clave de esta tabla se forma con el username del usuario y el nombre que se le de al diseño a la hora de almacenarlo.

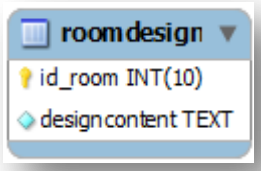


USERDESIGN		
Campo	Tipo	Descripción
username	*varchar(45)*	Nombre de usuario a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *username* de la tabla **User**.
designname	*varchar(45)*	Nombre del diseño con el que el usuario realiza el guardado. Este campo y el nombre de usuario forman la clave primaria de esta tabla.
designcontent	*text*	En este campo se guarda el contenido del lienzo que el usuario quiere guardar en forma de conjunto de bytes.

Tabla 67: Tabla UserDesign

- *Tabla **RoomDesign**:*

En esta tabla se almacenarán los diseños asociados a cada una de las salas. El sistema se ha desarrollado para que unicamente pueda existir como mucho un diseño asociado a la sala, es por ello que la clave primaria de esta tabla sea el id de la sala.



USERDESIGN		
Campo	Tipo	Descripción
id_room	*int(10)*	Id de la sala a la cual hace referencia esta relación. Es una clave ajena (Foreign Key) que hace referencia al campo *id* de la tabla **Room**.
designcontent	*text*	En este campo se guarda el contenido del lienzo que el usuario quiere guardar en forma de conjunto de bytes.

Tabla 68: Tabla RoomDesign

En el Anexo VI de este documento se adjuntan el script sql de creación de los distintos elementos de la base de datos (tablas, índices y datos iniciales).

Capa de control

En este apartado describiremos la capa de control del sistema, donde veremos el modelo de negocio seguido a la hora de implementar las funcionalidades del sistema. La descripción detallada de todas las clases de la capa de control se adjunta en este documento en el Anexo V.

La capa de control se encuentra en el servidor del sistema, y se comunica con la parte cliente del mismo de la siguiente forma.

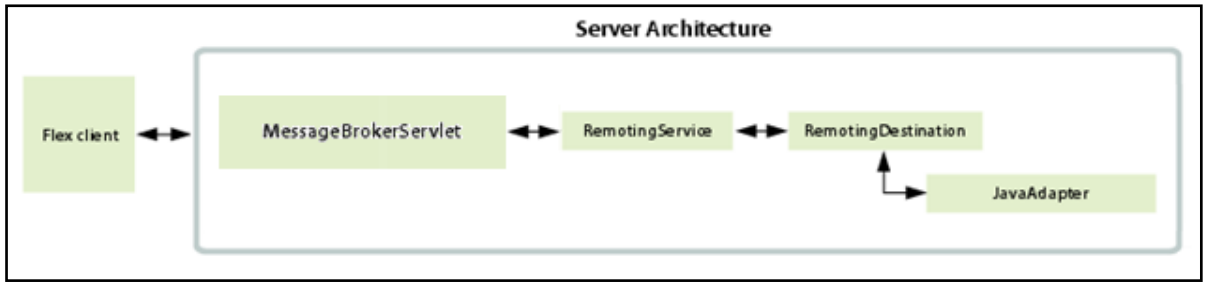


Figura 22: Comunicación Cliente - Servidor

A continuación se muestra el diagrama de clases diseñado de forma detallada.

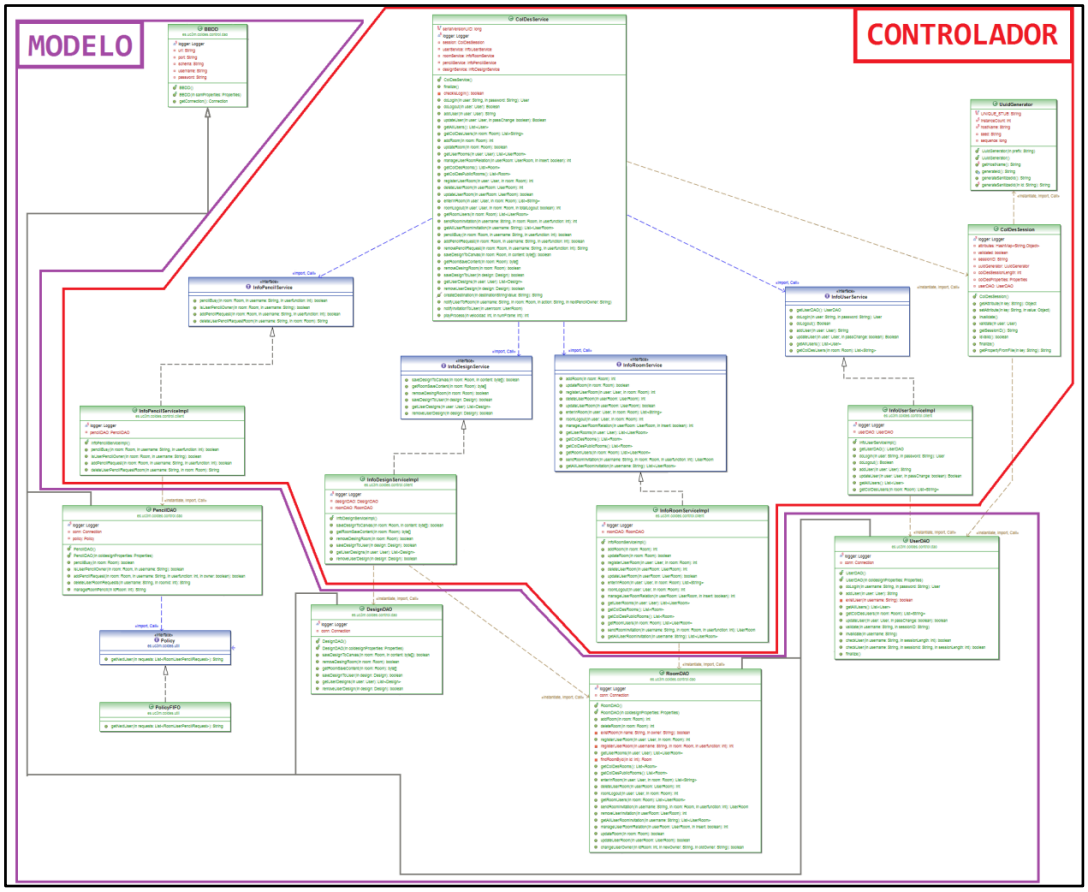


Figura 23: Diagrama de clases general

Antes de explicar el diagrama de clases representado en la figura anterior, es necesario hacer una introducción de las distintas clases creadas para modelar los distintos elementos del sistema, para de esta forma tener una mejor noción sobre el tipo de elementos que devuelven y reciben las distintas funciones del sistema.

El modelado de estos elementos se ha realizado mediante la creación de distintos **POJOS** que hacen referencia a cada uno de los elementos del sistema. **POJO** es un acrónimo de *Plain Old Java Objects*. El nombre se usa para enfatizar que el objeto dado es un objeto Java normal, no un objeto especial. En nuestro caso, en estas clases únicamente se definen los atributos y los métodos para poder accederlos y modificarlos (get y set).

Los POJOS definidos para el sistema son los siguientes:

- ***Clase User***: Mediante esta clase modelamos lo que sería un usuario.
- ***Clase Room***: Mediante esta clase modelamos lo que sería una sala para el sistema.
- ***Clase UserRoom***: Esta clase pretende modelar la relación entre salas y usuarios. Estas relaciones lo que indican es que usuarios están presentes en que salas.
- ***Clase RoomUserPencilRequest***: Esta clase modela una petición de un pincel por parte de un usuario en una determinada sala.
- ***Clase Design***: Esta clase modela lo que es un diseño dentro del sistema. Este diseño en el modelo actual es el que se usa para representar los diseños almacenados por los usuarios dentro del sistema. El diseño asociado a la sala se almacena directamente como un array de bytes en base de datos.

Una vez explicadas las clases de los elementos del sistema, podemos pasar a explicar cada uno de los componentes representados en la Figura 23.

Controlador

La parte controladora tiene cinco grandes bloques. Por un lado tenemos la clase que sirve de enlace entre la vista y el controlador del sistema, y por otro lado tenemos los cuatro módulos que la herramienta ha de controlar: usuarios, salas, pinceles y diseños.

La clase *ColDesService.java* que sirve de enlace entre la vista y el controlador dentro de sistema. En esta clase están definidos todos los métodos que la vista puede necesitar para ofrecer al usuario los servicios que la aplicación ha de satisfacer. Cada uno de estos métodos hará uso de un *service* distinto dependiendo de a que módulo haga referencia dicha función.

Esta clase es una del más importantes dado que, como hemos comentado con anterioridad, es un service que se encarga de ofrecer toda la funcionalidad al cliente de la aplicación. Es por ello que a continuación se muestra una descripción detallada de la clase. El resto de *services* del sistema específicos de los módulos se explican con detalle en el Anexo V de este documento.

Nombre	ColDesService.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	session	*ColDesSession*	Variable que gestiona las sesiones del sistema	
	userService	*InfoUserService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con los usuarios	
	roomService	*InfoRoomService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con las salas	
	pencilService	*InfoPencilService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con los diseños del sistema, tanto de usuarios como de salas.	
	designService	*InfoDesignService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con los diseños del sistema, tanto de usuarios como de salas	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	ColDesService	*Constructor*	-	
	Descripción			
	Constructor por defecto encargado de inicializar la sesión y los distintos service del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	finalize	*void*	-	
	Descripción			
	Función encargada de finalizar la sesión del DAO del usuario.			
	Nombre	*Tipo*	*Argumentos*	
	checkIsLogIn	*Boolean*	-	
	Descripción			
	Función que verifica si el usuario aún tiene una sesión valida dentro del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	doLogin	*User*	*String* user *String* password	
	Descripción			
	Función encargada de realizar la autenticación del usuario dentro del sistema. Para ello recurre al Service de usuarios.			
	Nombre	*Tipo*	*Argumentos*	
	doLogout	*Boolean*	*User* user	
	Descripción			
	Función encargada de finalizar la sesión del usuario autenticado en el sistema. Para ello invalida la sesión, y usa el Service de usuarios para finalizar las acciones pendientes del usuario en el sistema.			
	Nombre	*Tipo*	*Argumentos*	
	addUser	*String*	*User* user	

	Descripción		
	Añade un nuevo usuario al sistema mediante el Service de usuarios.		
	Nombre	*Tipo*	*Argumentos*
	updateUser	*Boolean*	*User* user *boolean* passChange
	Descripción		
	Actualiza los datos de un usuario mediante el Service de usuarios. En caso de cambio de contraseña se pasa a true el *boolean* passChange para indicar que hay que cambiar la contraseña del usuario.		
	Nombre	*Tipo*	*Argumentos*
	getAllUsers	*List <User>*	-
	Descripción		
	Función que obtiene todos los usuarios del sistema mediante el Service de usuarios.		
	Nombre	*Tipo*	*Argumentos*
	getColDesUsers	*List <String>*	*Room* room
	Descripción		
	Función que devuelve todos los usuarios que están en una determinada sala en el momento actual.		
	Nombre	*Tipo*	*Argumentos*
	addRoom	*int*	*Room* room
	Descripción		
	Añade una nueva sala al sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	updateRoom	*boolean*	*Room* room
	Descripción		
	Actualiza los datos referentes de una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getUserRooms	*List <UserRoom>*	*User* user
	Descripción		
	Obtiene la lista de salas en las cuales está participando un usuario, así como la función que desempeña en cada una de las salas.		
	Nombre	*Tipo*	*Argumentos*
	manageUserRoomRelation	*int*	*UserRoom* userRoom *boolean* insert
	Descripción		
	Función encargada de gestionar las relaciones de los usuarios con las distintas salas del sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getColDesRooms	*List <Room>*	-
	Descripción		
	Obtiene las distintas salas del sistema sin excepción mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getColDesPublicRooms	*List <Room>*	-
	Descripción		
	Obtiene las salas públicas del sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	registerUserRoom	*int*	*User* user *Room* room
	Descripción		

	Asocia un usuario a una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	deleteUserRoom	*int*	*UserRoom* userRoom
	Descripción		
	Borrar un usuario de una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	updateUserRoom	*Boolean*	*UserRoom* userRoom
	Descripción		
	Actualiza una relación usuario - sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	enterInRoom	*List <String>*	*User* user *Room* room
	Descripción		
	Función mediante la cual se notifica al sistema la entrada de un usuario a una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	roomLogout	*int*	*User* user *Room* room *Boolean* totalLogout
	Descripción		
	Función que notifica al sistema cuando un usuario sale de la sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getRoomUsers	*List <UserRoom>*	*Room* room
	Descripción		
	Obtiene las distintas relaciones que tiene una sala con los usuarios del sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	sendRoomInvitation	*int*	*String* username *Room* room *int* userfunction
	Descripción		
	Función mediante la cual se le envía una invitación a sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getAllUserroomInvitation	*List <UserRoom>*	*String* username
	Descripción		
	Función mediante la cual se obtienen todas las invitaciones que hay en el sistema de un usuario mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	pencilBusy	*Boolean*	*Room* room *String* username *int* userfunction
	Descripción		
	Función que verifica si el pincel está siendo usado en una sala mediante el Service de pinceles.		
	Nombre	*Tipo*	*Argumentos*
	addPencilRequest	*Boolean*	*Room* room *String* username *int* userfunction
	Descripción		

	Función mediante la cual un usuario realiza una petición para poder usar el pincel mediante el Service de pinceles. Adicionalmente se manda un mensaje a todos los usuarios de la sala para notificar la del pincel.		
	Nombre	*Tipo*	*Argumentos*
	removePencilRequest	*String*	*Room* room *String* username *int* userfunction
	Descripción		
	Función mediante la cual se elimina una petición de un pincel de una sala mediante el Service de pinceles.		
	Nombre	*Tipo*	*Argumentos*
	saveDesignToCanvas	*Boolean*	*Room* room *Byte[]* content
	Descripción		
	Función encargada de asociar un diseño a una sala mediante el Service de diseños.		
	Nombre	*Tipo*	*Argumentos*
	getRoomSaveContent	*byte[]*	*Room* room
	Descripción		
	Obtiene el diseño asociado a una sala mediante el Service de diseños.		
	Nombre	*Tipo*	*Argumentos*
	removeDesingRoom	*Boolean*	*Room* room
	Descripción		
	Función encargada de borrar un diseño de una sala mediante el Service de diseños.		
	Nombre	*Tipo*	*Argumentos*
	saveDesignToUser	*Boolean*	*Design* design
	Descripción		
	Guarda para un usuario un diseño mediante el Service de diseños.		
	Nombre	*Tipo*	*Argumentos*
	getUserDesigns	*List <Design>*	*User* user
	Descripción		
	Función que devuelve los distintos diseños asociados a un usuario mediante el Service de diseños.		
	Nombre	*Tipo*	*Argumentos*
	removeUserDesign	*Boolean*	*Design* design
	Descripción		
	Borra un diseño asociado a un usuario mediante el Service de diseños.		
	Nombre	*Tipo*	*Argumentos*
	createDestination	*String*	*String* destinationStringValue
	Descripción		
	Crea canales que usara la aplicación para sincronizar a los distintos usuarios mediante los chats y lienzos de la sala. Este método en la última versión esta *deprecated*.		
	Nombre	*Tipo*	*Argumentos*
	notifyUserToRoom	*void*	*String* username *Room* room *String* action *String* nextPencilOwner
	Descripción		

	Función mediante la cual se notifica de distintas acciones en una sala al resto de usuarios de la sala en cuestión.		
	Nombre	*Tipo*	*Argumentos*
	notifyInvitationToUser	*void*	*UserRoom userroom*
	Descripción		
	Función mediante la cual se notifica al usuario en tiempo real de la invitación a una sala.		
	Nombre	*Tipo*	*Argumentos*
	playProcess	*int*	*int* velocidad *int* numFrame
	Descripción		
	Función encargada de simular el *play* de acciones de la sala.		

Tabla 69: Clase ColDesService

Por otro lado, para el gestionar y controlar de las sesiones de los usuarios dentro del sistema, se han implementado las clases *ColDesSession.java* y *UuidGenerator.java*. La clase *ColDesSession.java* es la encargada de modelar una sesión válida dentro del sistema, en el momento que esta sesión se invalida, el usuario asociado deja de tener permisos dentro de la aplicación para seguir realizando acciones en la misma. La sesión de un usuario tiene una duración máxima de 30 minutos y se puede invalidar en el caso de que el usuario autenticado se autentique de nuevo en el sistema desde el mismo equipo o desde otro cualquiera. Por otro lado, la clase *UuidGenerator.java* únicamente se encarga de generar un id de sesión basándose en el host del usuario que inicia sesión y en el momento en el cual inicia dicha sesión.

Todos los *services* del sistema, tal y como hemos comentado anteriormente, son usados por la clase *ColDesService.java* para la invocación de los distintos métodos de los módulos del sistema mediante el uso de interfaces.

Los *service* específicos de esta aplicación se encuentran implementados en clases definidas con el sufijo *Impl*. De esta forma tenemos definidos en el sistema los siguientes *services* y sus correspondientes implementaciones para cumplir las necesidades requeridas por el sistema:

- **InfoUserService.java** → *InfoUserServiceImpl.java*
- **InfoRoomService.java** → *InfoRoomServiceImpl.java*
- **InfoPencilService.java** → *InfoPencilServiceImpl.java*
- **InfoDesignService.java** → *InfoDesignServiceImpl.java*

Como ya adelantábamos anteriormente, la explicación detallada de cada una de estas clases se encuentra en el Anexo V de este documento, a continuación se explicara de forma breve el funcionamiento y la función que desempeña cada uno de los *service* anteriormente expuestos.

El *service* de usuarios es el encargado de gestionar todas las peticiones que tienen que ver con el módulo de usuarios del sistema. Este *service* es usado desde la capa más externa del controlador del sistema y se basa por otro lado en la parte del modelo encargada de gestionar los usuarios dentro de la base de datos mediante la clase ***UserDAO***.

El *service* de salas es el encargado de gestionar todas las peticiones que tienen que ver con el módulo de salas del sistema. Este *service* es usado desde la capa más externa del controlador del sistema y se basa por otro lado en la parte del modelo encargado de gestionar las salas, las relaciones con los usuarios y las invitaciones a salas dentro de la base de datos mediante la clase ***RoomDAO***.

El *service* de pinceles es el encargado de gestionar los pinceles de las salas en las cuales la participación se realiza por turnos. Es el encargado de gestionar todas las peticiones de pinceles de los usuarios, de notificar las peticiones al resto de usuarios de las salas y controlar si el pincel tiene o no propietario. Este *service* es usado desde la capa más externa del controlador del sistema y se basa por otro lado en la parte del modelo encargado de gestionar los pinceles dentro de la base de datos mediante la clase ***PencilDAO***.

Por último, el *service* de diseños es el encargado de gestionar los diseños del sistema. Se encarga de asociar diseños a salas y a usuarios. Una sala únicamente puede tener asociado en todo momento un diseño. Por otro lado, un usuario a la hora de guardarse un diseño ha de especificar un nombre para dicho diseño, y este puede almacenar tantos diseños como desee. Este *service* es usado desde la capa más externa del controlador del sistema y se basa por otro lado en la parte del modelo encargado de gestionar los diseños dentro de la base de datos mediante la clase ***DesignDAO***. También se basa en el DAO de salas (***RoomDAO***) para poder asociar a una sala un diseño dado.

Modelo

El modelo del sistema se centra en los cuatro grandes módulos que el sistema ha de gestionar y controlar: usuarios, salas, pinceles y diseños.

Las clases diseñadas para llevar este control heredan todas de una superclase que define los métodos básicos de conexión en base de datos y de carga de parámetros mediante un fichero de configuración:

Nombre	BBDD.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	url	*String*	URL de la base de datos.	
	port	*String*	Puerto en el que se encuentra escuchando la base de datos.	
	schema	*String*	Schema que contiene la estructura de tablas de la aplicación.	
	username	*String*	Nombre de usuario para conectarse a la base de datos.	
	password	*String*	Contraseña del usuario de la base de datos.	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	BBDD	*Constructor*	-	
	Descripción			
	Constructor por defecto en el que se leen las propiedades de la base de datos del fichero de propiedades *coldes.properties.*			
	Nombre	*Tipo*	*Argumentos*	
	BBDD	*Contructor*	*Properties* ColDesProperties	
	Descripción			
	Constructor que recibe ya las distintas propiedades de la conexión a base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	getConnection	*Connection*	-	
	Descripción			
	Función mediante la cual se obtiene una conexión a la base de datos del sistema.			

Tabla 70: Clase BBDD

El fichero de configuración que usa esta clase para cargar los datos referentes a la base de datos se denomina en el sistema *coldes.properties* y sigue la siguiente estructura:

```
####################################
# Archivo de propiedades de ColDes #
####################################

# Duración de la sesión, medido en minutos
session.length=30

# URL de la máquina
server.url=localhost

# Acceso a la base de datos
db.url=localhost
db.port=3306
db.schema=coldesbd
db.username=coldesbd
db.password=coldesbd
```

De forma general, los distintos **DAO's** (*Data Access Object*) definidos en el sistema seguirían la siguiente estructura de clases:

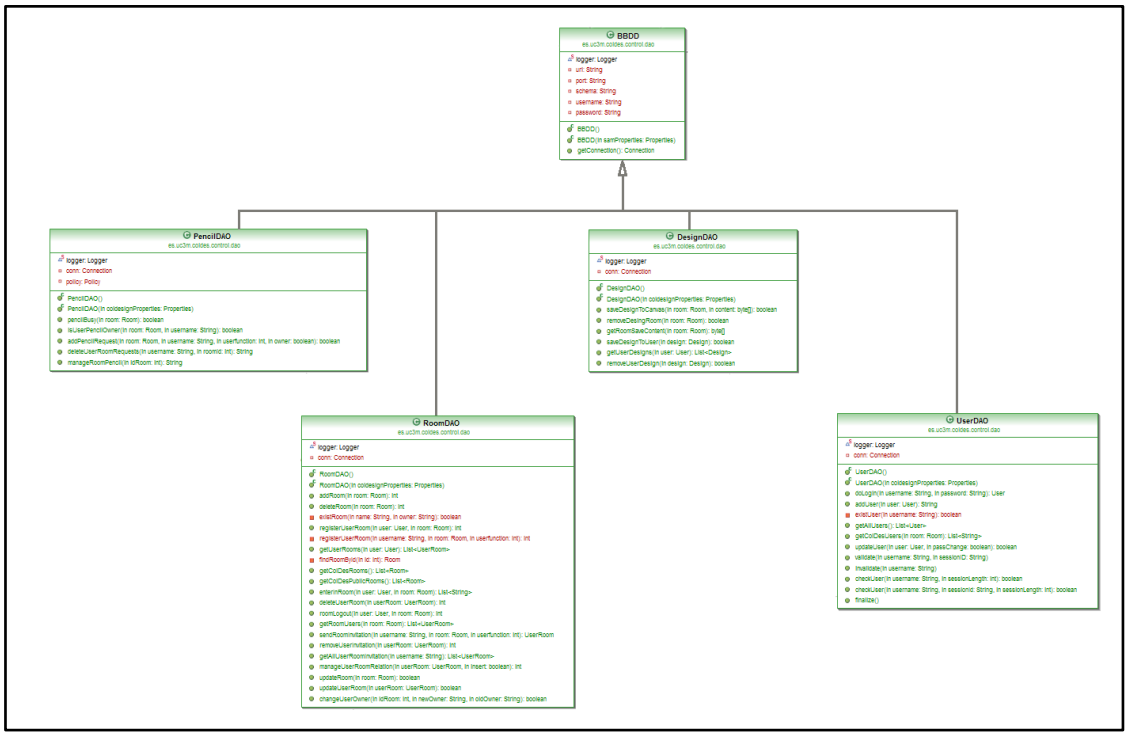


Figura 24: Estructura de clases del modelo

A continuación iremos viendo de forma resumida una a una las clases creadas para gestionar la parte del modelo de los elementos del sistema: usuarios, salas, diseños y pinceles. La descripción detallada de cada una de estas clases, al igual que pasaba con el código del controlador, se adjunta en el Anexo V de este documento.

El *DAO* de usuarios es usado desde el *service* de usuarios, ***InfoUserServiceImpl.***, y por la clase que modela la sesión dentro del sistema, ***CoDesSession***, de tal forma que para cada una de las acciones que el usuario realice dentro del sistema se pueda verificar si el usuario aún tiene la sesión activa.

El *DAO* de salas es usado desde el *service* de salas, ***InfoRoomServiceImpl***, y por otro parte para asociar diseños a la sala también es usado por el *service* de diseños, ***InfoDesignServiceImpl***.

El *DAO* de pinceles se usa desde el *service* de pinceles, ***InfoPencilServiceImpl***. Este *DAO* es el encargado de gestionar todo lo referente a las distintas peticiones de pinceles de las salas y de asignar el pincel siguiendo una política de asignación de pinceles que comentaremos más adelante.

La asignación de los pinceles de las distintas salas se realiza entre las distintas peticiones que el sistema ha procesado dentro de la sala. Para seleccionar el siguiente propietario del pincel dentro de cada sala, se debe usar una clase que implemente los métodos definidos en la interfaz *Policy*. El proyecto se ha desarrollado siguiendo por defecto una política FIFO, en la que la primera petición que se realiza es la primera petición que se atiende. En la siguiente tabla se presenta una descripción de la clase e interfaz comentadas:

Nombre	Policy.java / PolicyFIFO.java		**Tipo**	Interfaz /Clase
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	getNextUser	*String*	*List<RoomUserPencilRequest>* request	
	Descripción			
	Función que se ha de implementar encargada de seleccionar la siguiente petición de una lista de peticiones. En este caso la siguiente petición es la primera de la lista que se le pasa, o *null* en caso de no existir peticiones.			

Tabla 71: Interfaz/Clase Policy.java/PolicyFIFO.java

Por último, el *DAO* de diseños es usado desde el *service* de diseños, ***InfoDesignServiceImpl***. Mediante este DAO se gestionan los diseños de la aplicación, tantos los referentes a las salas como los referentes a los usuarios del sistema.

Capa de presentación

En este apartado trataremos todo lo referente al desarrollo de la capa de presentación del sistema, la parte cliente, desde el sistema de navegación entre las distintas pantallas del sistema, los distintos prototipos por lo que ha pasado el sistema y por ultimo finalizaremos mostrando el sistema final desarrollado.

Diagrama de navegación de las interfaces

En este apartado se dará una visión global del sistema proporcionado una navegación entre las diferentes interfaces que forman el sistema.

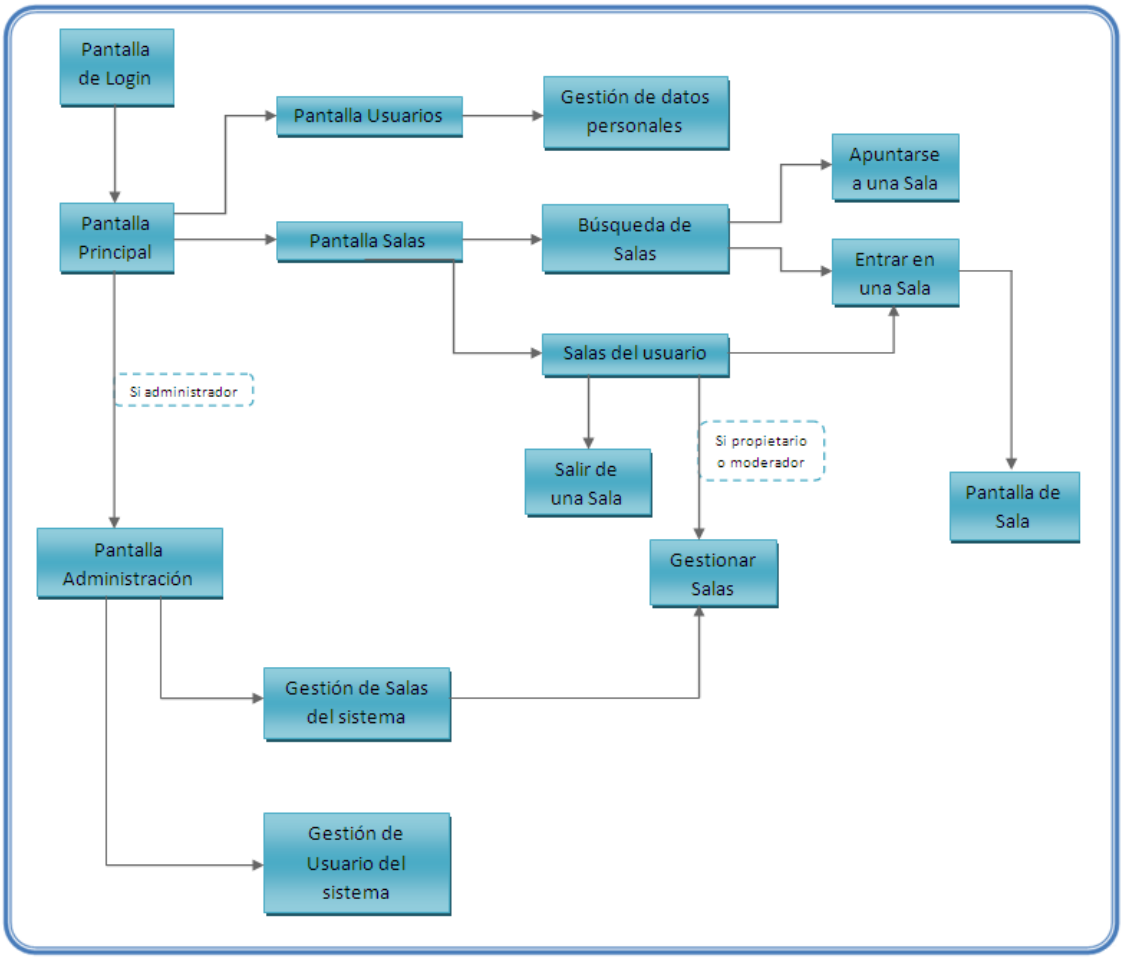


Figura 25: Diagrama de Navegación

Como se puede observar en el diagrama, todo el flujo de navegación comienza en la pantalla de autenticación del sistema, en la cual el usuario deberá administrar al sistema su nombre de usuario y contraseña. Tras esto, el usuario accederá a la pantalla principal del sistema, desde la cual podrá acceder al resto de interfaces del sistema. En este punto, en caso de ser administrador del sistema, el usuario también podrá acceder a la interfaz de administración, desde la cual se pueden gestionar todos los elementos del sistema: usuarios y salas.

Otro punto interesante del diagrama de navegación es la gestión de salas. Esta gestión se podrá realizar desde dos puntos distintos del sistema:

- Si el usuario tiene perfil administrador, podrá gestionar los datos de todas las salas del sistema (usuarios de la misma y datos de la sala) sin excepción desde el menú de administración del sistema.
- Desde la pantalla de salas personales, el usuario podrá realizar acciones sobre la sala, dependiendo de la función que desempeña en cada una de ellas:
    - Invitado: No puede gestionar.
    - Colaborador: No puede gestionar.
    - Moderador: Puede gestionar los usuarios de la sala.
    - Propietario: Puede gestionar los datos de la sala y los usuarios de la misma.

Por último, en la pantalla de sala, el usuario podrá acceder al lienzo y al chat definidos para la misma, y realizar acciones acordes con la función que desempeña en la misma.

Prototipos

El desarrollo de la parte visual del cliente se ha realizado mediante la implementación de distintos prototipos. A continuación mostramos como ha ido evolucionando este diseño hasta llegar al diseño elegido finalmente.

La idea fundamental del proyecto era la existencia de una pizarra mediante la cual los distintos usuarios del sistema pudiesen colaborar de forma gráfica. Es por ello que el primer prototipo que se desarrollo fue una pizarra muy básica:

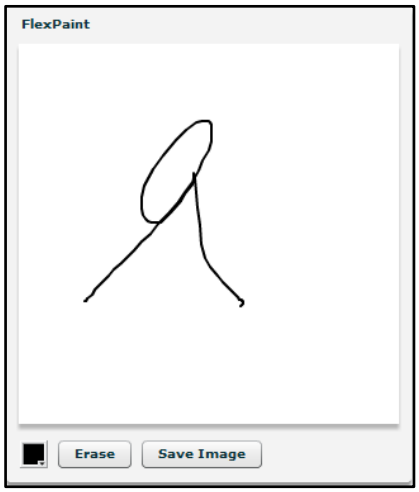


Figura 26: Prototipo - Lienzo básico

En esta versión el usuario básicamente podía hacer:

- Dibujar trazos.
- Definir un color para el trazo.
- Guardar el contenido del lienzo a una imagen en su PC.
- Limpiar el contenido del lienzo.

Una vez desarrollado un componente simple que permitía a los usuarios pintar sobre un lienzo, se paso a desarrollar un prototipo de lienzo que no solo dejase dibujar trazas, sino que también permitiese dibujar otros elementos y realizar más acciones con el contenido del lienzo:

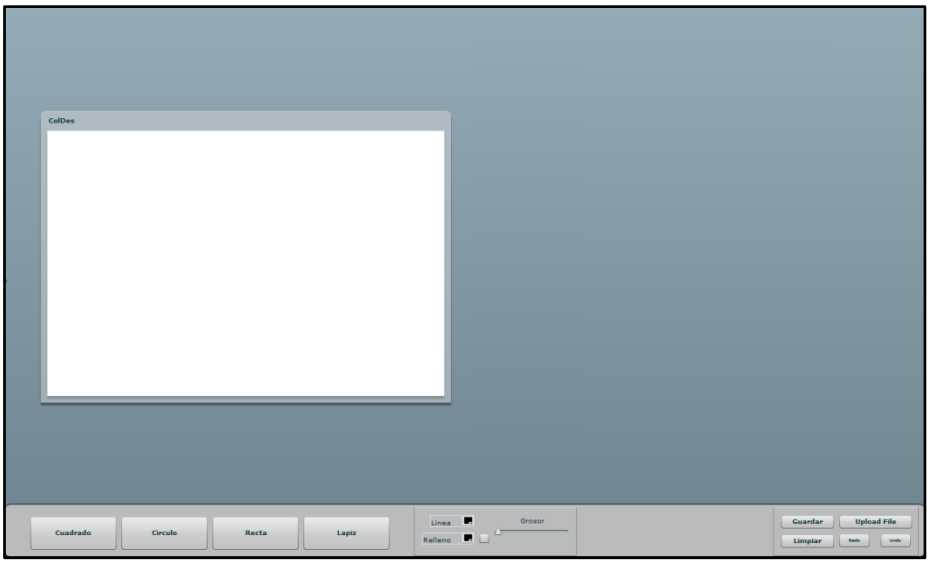


Figura 27: Prototipo - Lienzo avanzado I

Con esto, el usuario ya podía dibujar más objetos sobre el lienzo y con muchas más posibilidades frente al prototipo inicial. El usuario podía:

- Dibujar círculos, cuadrados, rectas y trazos.
- Definir un color y una anchura para los bordes de las figuras.
- Especificar un color de relleno para los cuadrados y los círculos.
- Hacer y deshacer las acciones sobre el lienzo.
- Guardar el contenido del lienzo a una imagen en su PC.
- Cargar imágenes desde el PC al lienzo.
- Limpiar el contenido del lienzo.

Después de esto, se vio la necesidad de incluir en el proyecto un componente que controlase el acceso a la aplicación por parte de los usuarios, naciendo así dentro del proyecto modulo de usuarios, donde los usuarios debían registrarse para crear un usuario y poder acceder a la aplicación:

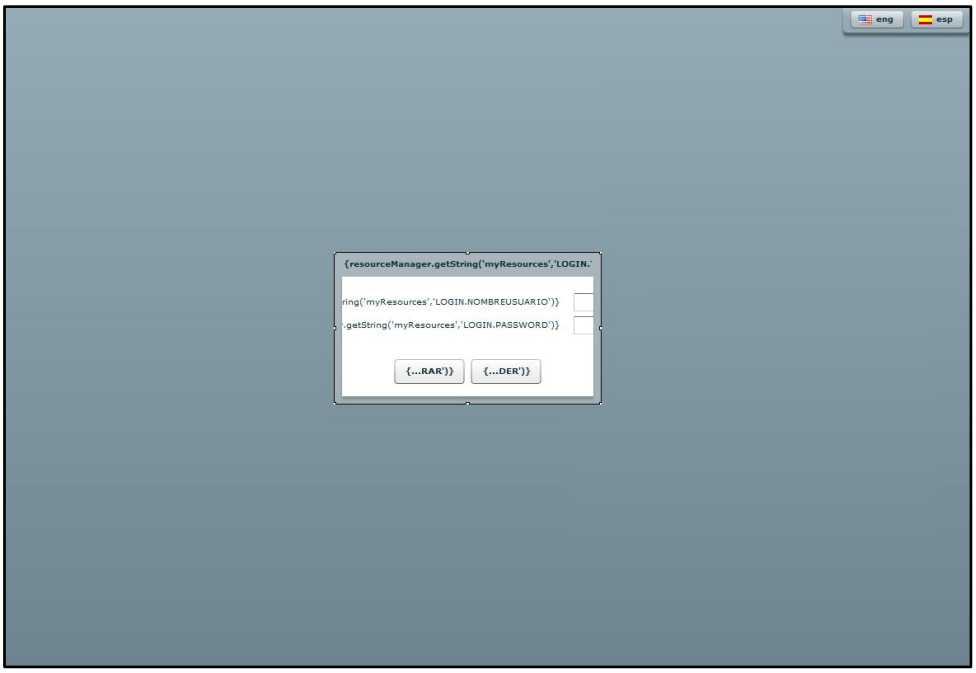


Figura 28: Prototipo - Modulo de autenticación

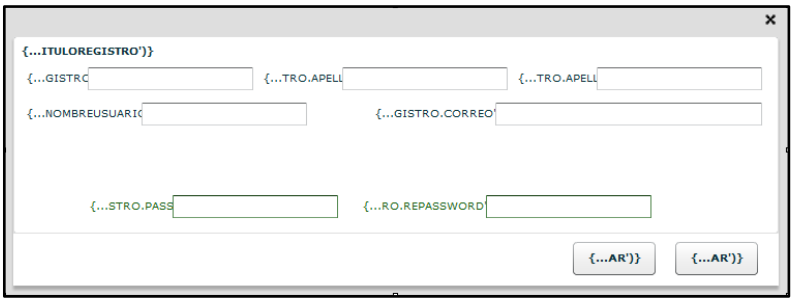


Figura 29: Prototipo - Formulario de registro para usuarios

De forma adicional, se modifico el formato del lienzo pero sin modificar las funcionalidades que este ya ofrecía:

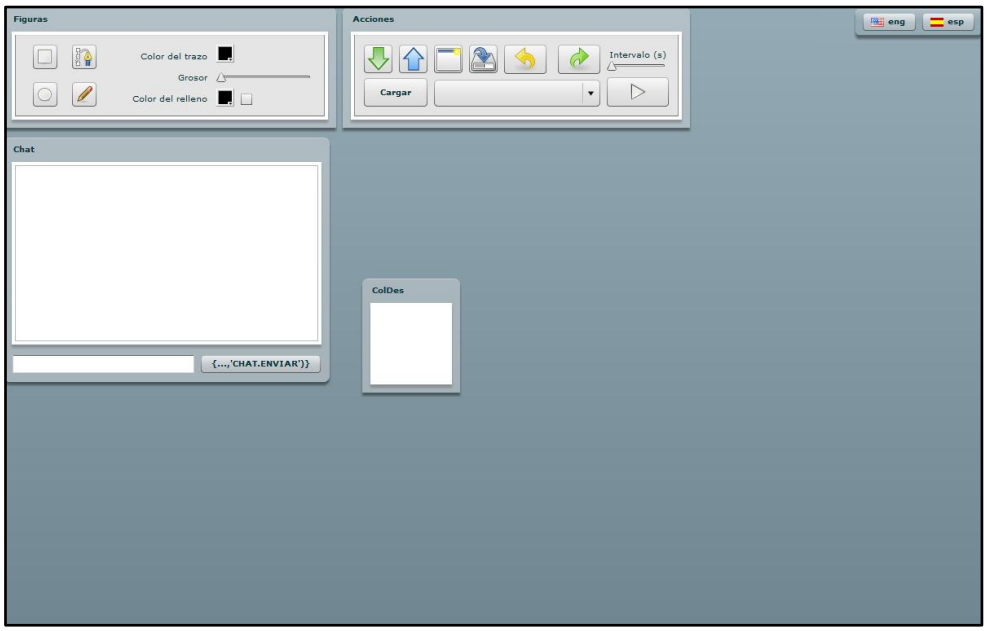


Figura 30: Prototipo - Lienzo avanzado II

A partir de este prototipo se empezó a pensar en el sistema como un conjunto de salas en el que cada una de ellas tendría su propio lienzo y chat. De esta forma surgió el siguiente prototipo, donde ya se le dio forma a los dos grandes módulos que componen el sistema, por una lado el de usuarios y por otro el de salas.

En este nuevo prototipo el usuario tendría distintos perfiles, administrador y diseñador dentro del sistema, y distintas funciones dentro de cada una de las salas.

El administrador podría gestionar los datos referentes a los usuarios, ver los distintos usuarios del sistema y crear nuevos usuarios:

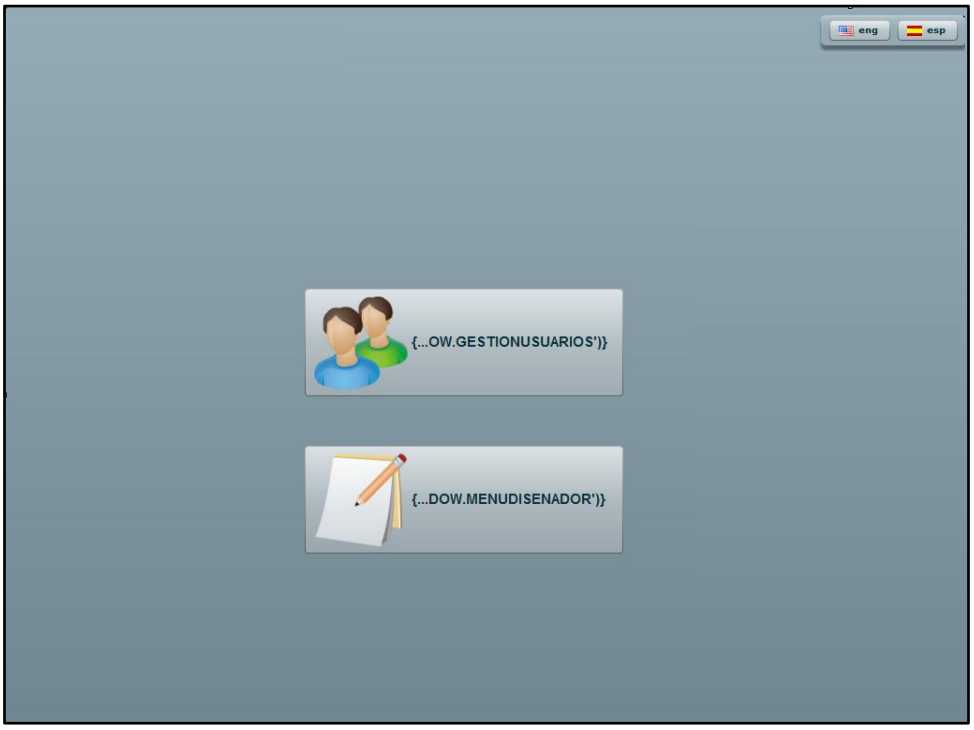


Figura 31: Prototipo - Pantalla de Administración

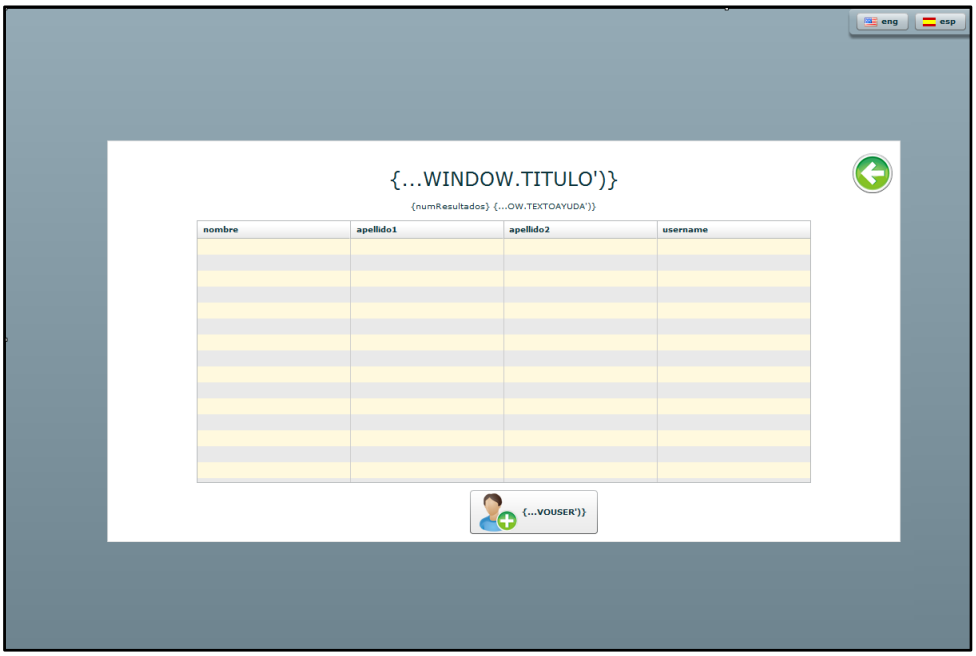


Figura 32: Prototipo - Pantalla de gestión de usuarios

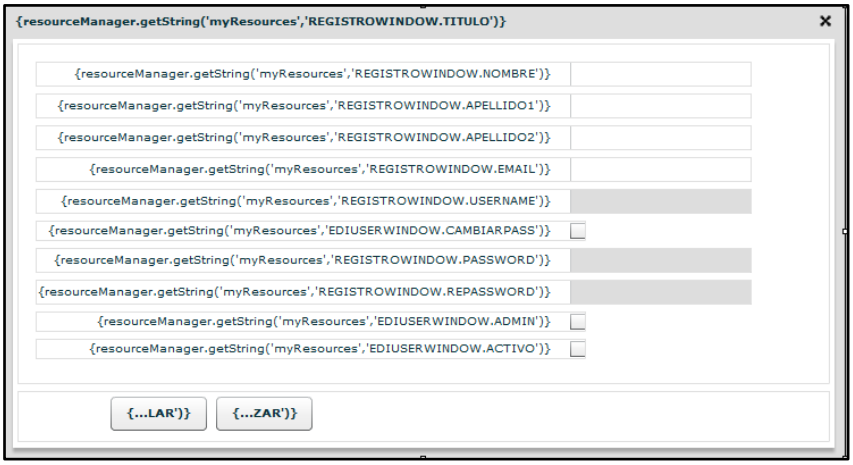


Figura 33: Prototipo - Formulario de registro de usuarios

Por otro lado, el perfil diseñador podría acceder al menú de salas, pudiendo crear, buscar, ver y acceder a las salas en las cuales está participando, donde las acciones que puede realizar van en función del rol que desempeñe en la sala.

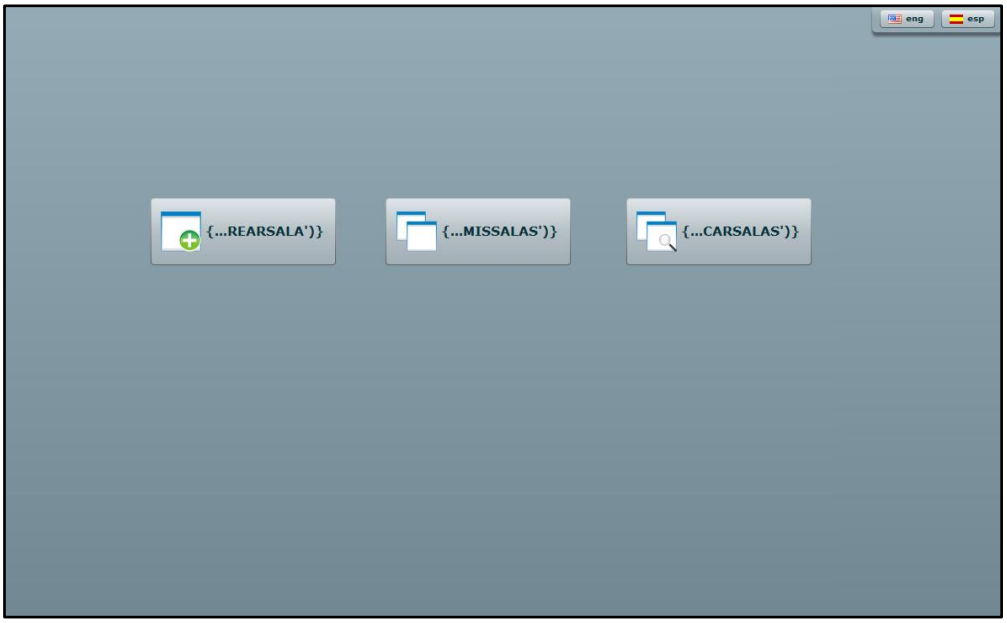


Figura 34: Prototipo - Pantalla Principal

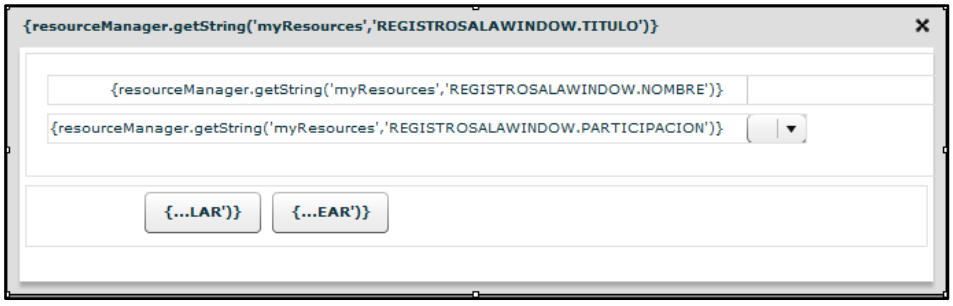


Figura 35: Formulario de creación de salas

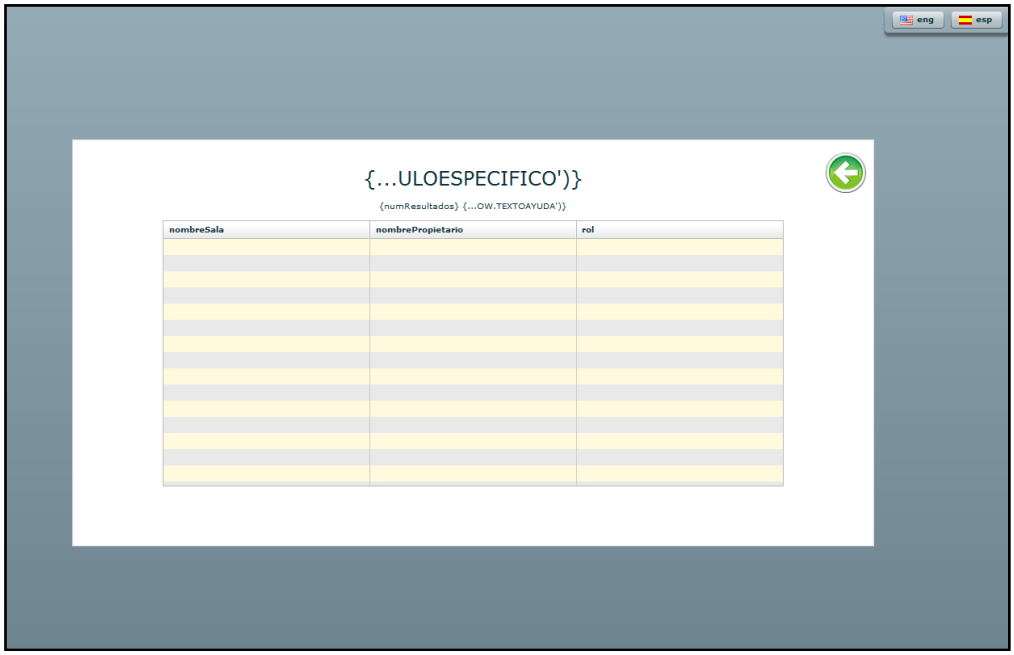


Figura 36: Prototipo - Pantalla de salas del usuario

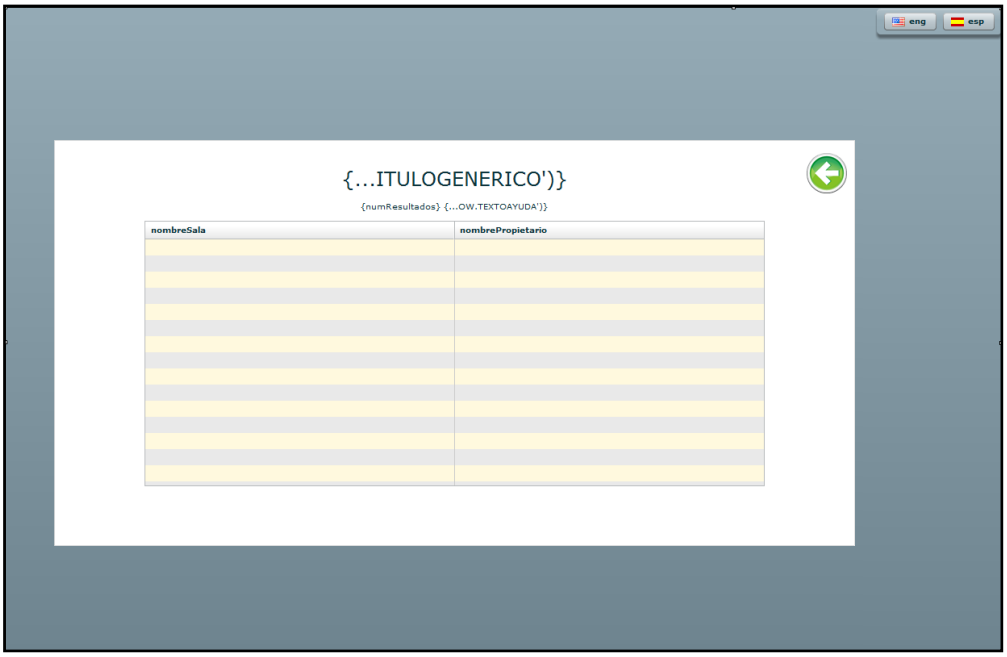


Figura 37: Prototipo - Pantalla de salas del sistema

Diseño final

A continuación veremos por encima la aplicación mediante una serie de figuras que muestren los cambios más relevantes de la aplicación con respecto a los prototipos desarrollados anteriormente.

Uno de los cambios más visuales que ha sufrido la aplicación han sido los colores de la misma. Se ha optado por cambiar el color por defecto de los componentes por un color más claro que fuese más atractivo visualmente.

Figura 38: Diseño Final - Pantalla de Autenticación

Otro cambio relevante de la aplicación es la gestión de las distintas pantallas de la aplicación. Con el diseño final de la aplicación se ha optado por trabajar con un sistema de pestañas para cada una de las ventanas de la aplicación.

Figura 39: Diseño Final - Pestañas de Ventanas

De esta forma los usuarios pueden acceder de forma más rápida y directa a los módulos de la aplicación. Con esta implementación, también se permite al usuario estar trabajando en distintas salas al mismo tiempo, pues cada una de las salas en las que entre aparecerán en la barra de ventanas, facilitando de esta forma el acceso a las mismas de forma considerable.

La pantalla principal de la aplicación se ha modificado también para mostrar de forma directa los menús principales y los submenús de cada uno de ellos en la misma ventana. En función del perfil del usuario, este tendrá una pantalla de inicio u otra:

El menú de inicio de un usuario normal será el siguiente:

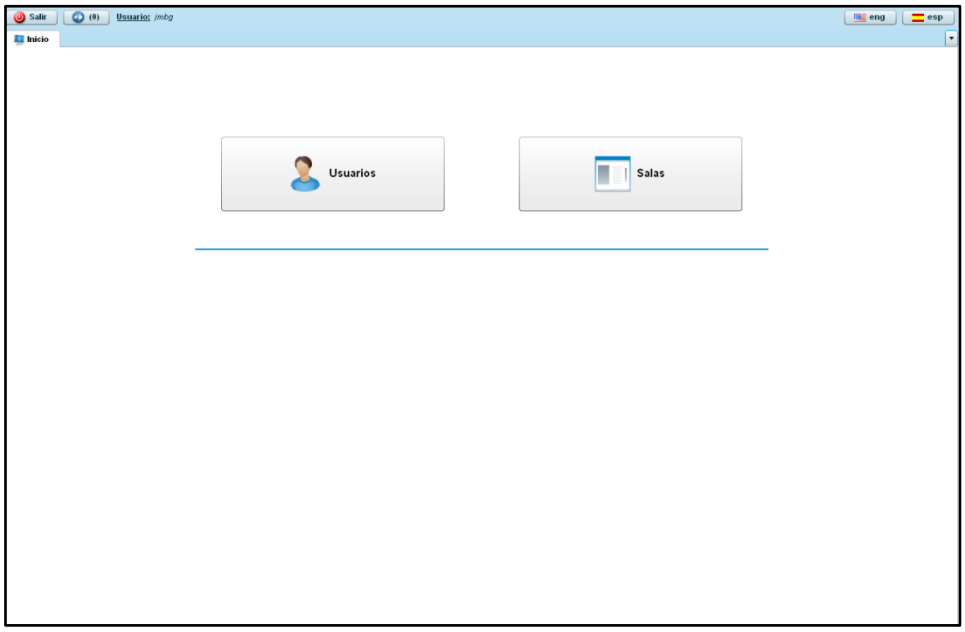


Figura 40: Diseño Final - Pantalla de Inicio usuario normal

Por otro lado el menú de inicio de un usuario administrador será:

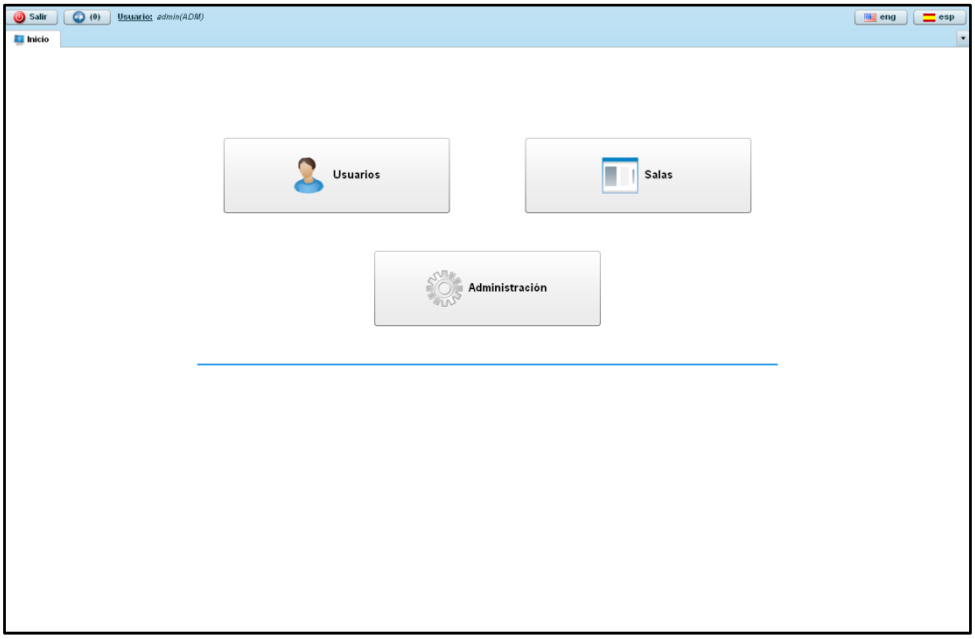


Figura 41: Diseño Final - Pantalla de Inicio usuario administrador

Cada uno de estos menús desplegara debajo de la barra los submenús asociados de la siguiente forma (visualizaremos los menús a partir de un usuario administrador para evitar repetir figuras).

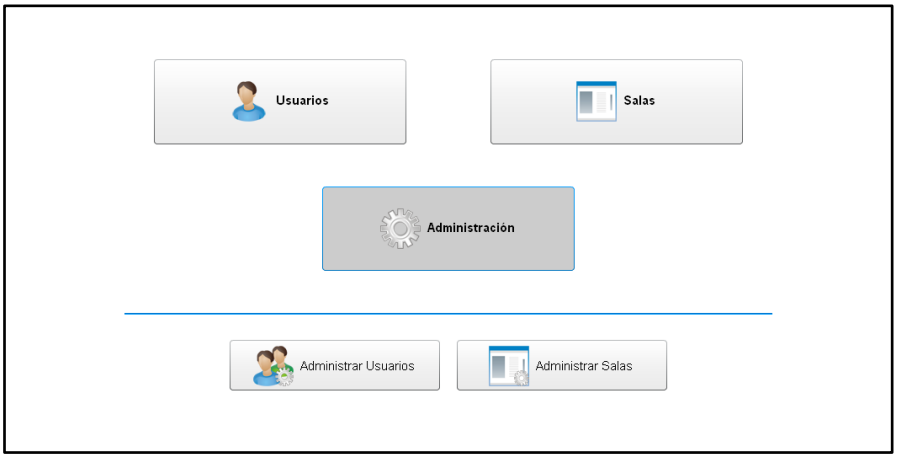


Figura 42: Diseño Final - Submenús Administración

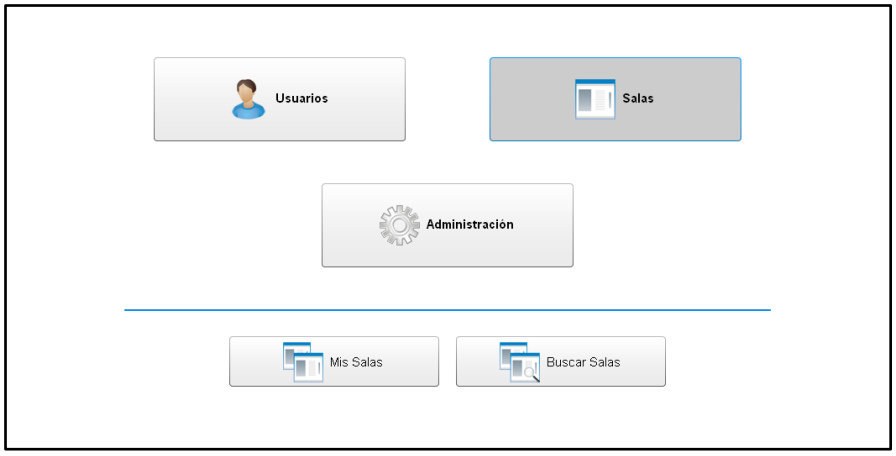


Figura 43: Diseño Final - Submenús Salas

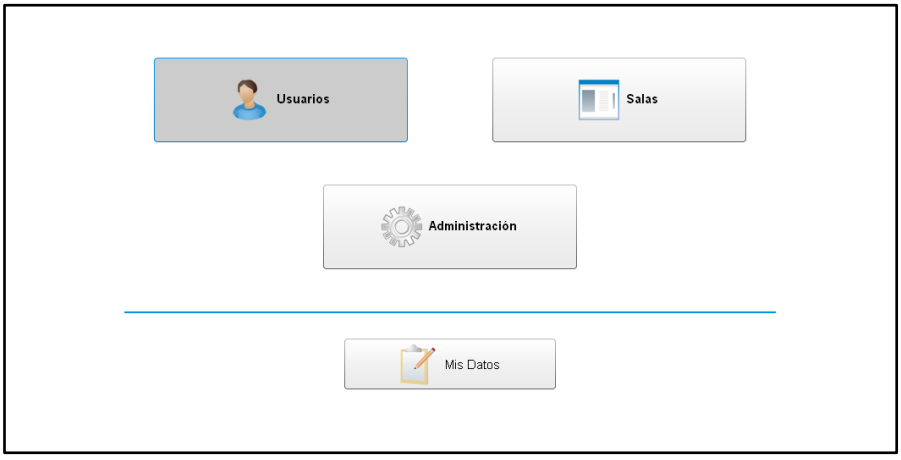


Figura 44: Diseño Final - Submenús Usuarios

Otro de los cambios introducidos en la versión definitiva del diseño es el formato de las ventanas de gestión de salas y usuarios, y de ventanas de búsqueda de salas personales y publicas del sistema.

La ventana de gestión de usuarios se ha modificado incluyendo una parte de búsqueda que permita localizar usuarios cuyo nombre o nombre de usuario sea semejante al introducido en el campo de búsqueda.

Figura 45: Diseño Final - Cuadro de Búsqueda

Otra modificación ha sido indicar visualmente si el usuario esta activo o no (verde si esta activo, rojo en caso contrario).

Figura 46: Diseño Final - Gestión de usuarios

Por otro lado, para facilitar la misión de los administradores, se ha creado una ventana nueva para la gestión de salas por parte de los administradores:

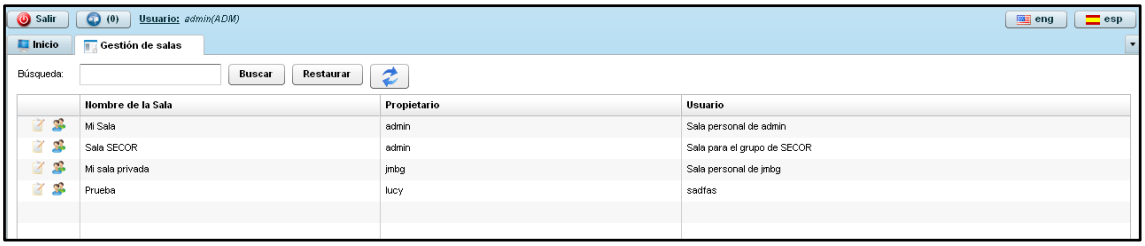


Figura 47: Diseño Final - Gestión de salas

En esta ventana los administradores pueden gestionar:

-  Cambiar los datos específicos de las salas.
-  Gestionar los usuarios de la sala: cambiar funciones de los usuarios, expulsar usuarios de la sala e invitar usuarios a participar en la sala con una determinada función.

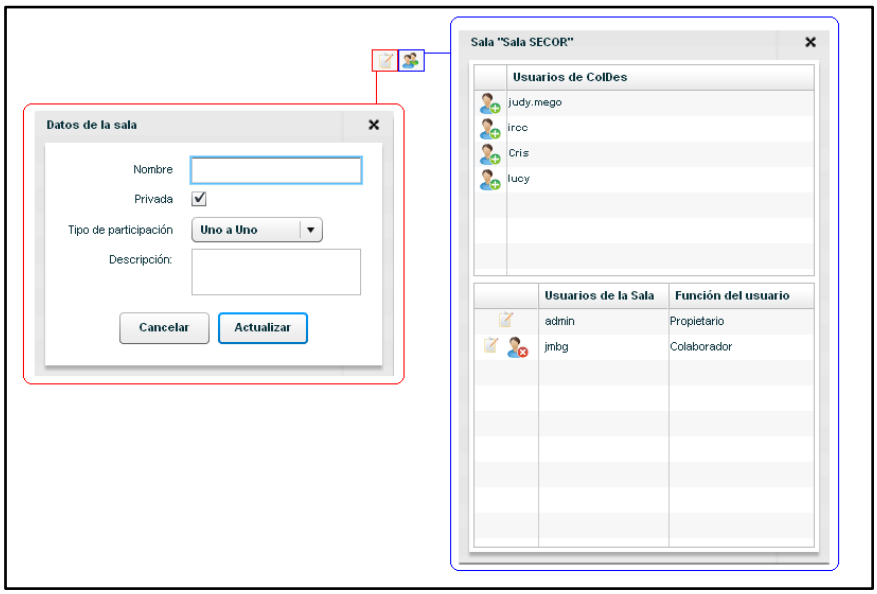


Figura 48: Diseño Final - Botones de gestión de salas

Por otro lado, se han introducido cambios también en las ventanas de búsqueda de salas ("*Mis Salas*" y "*Buscar Salas*"), en la cual ahora se permite realizar búsquedas de salas por nombre de sala o de usuario propietario, de igual forma que veíamos anteriormente en la pantalla de gestión de usuarios (*Figura: Diseño Final - Cuadro de Búsqueda*).

En la ventana de "*Mis Salas*", el usuario podrá gestionar las distintas salas en las cuales está participando, y el nivel de gestión que este podrá tener sobre estas salas dependerá de la función que este desempeña dentro de la sala. De forma esquemática, por función el usuario podrá:

- Propietario:
  -  Cambiar los datos específicos de las salas.
  -  Gestionar los usuarios de la sala: cambiar funciones de los usuarios, expulsar usuarios de la sala e invitar usuarios a participar en la sala con una determinada función.
  -  Borrar la relación del usuario con la sala. Esta acción conllevara en este caso el borrado de la sala y con ello, el borrado de todas las relaciones de los usuarios con la sala.

- Moderador:
  -  Gestionar los usuarios de la sala: cambiar funciones de los usuarios, expulsar usuarios de la sala e invitar usuarios a participar en la sala con una determinada función.
  -  Borrar la relación del usuario con la sala.
- Colaborador / Invitado:
  -  Borrar la relación del usuario con la sala.

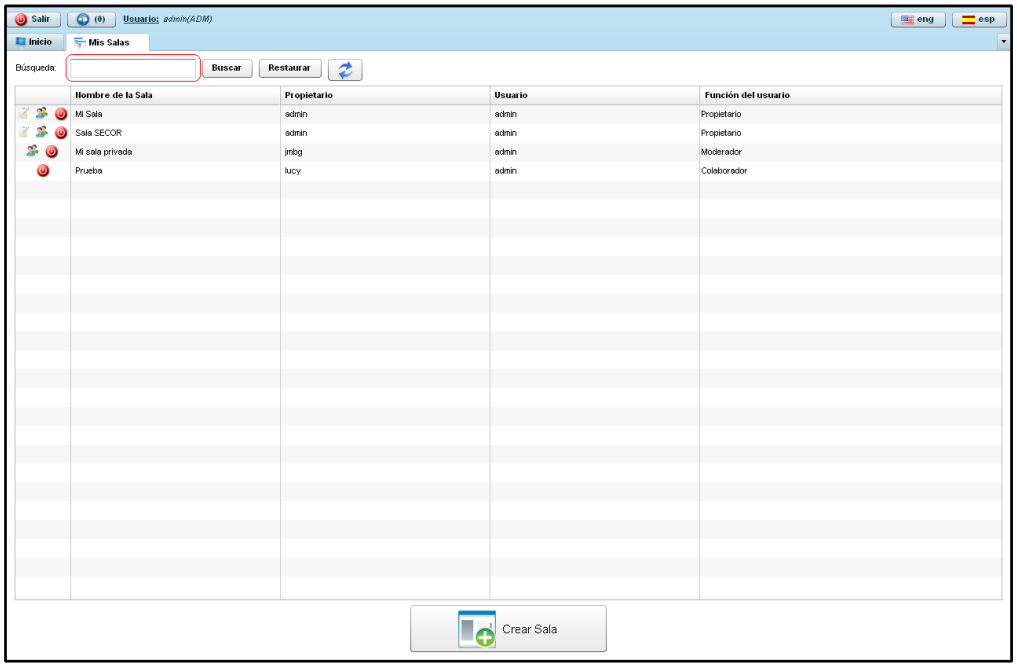


Figura 49: Diseño Final - Ventana Mis Salas

Además, el usuario desde esta ventana tendrá la opción de crear salas nuevas mediante el botón de la parte inferior de la pantalla "Crear Sala"

Figura 50: Diseño Final – Formulario de nueva sala

En la ventana de "*Buscar Salas*" el usuario podrá encontrar las salas públicas que hay creadas en el sistema, y a través de esta podrá entrar a participar en las mismas haciendo doble click sobre la sala en concreto a la cual quiere entrar.

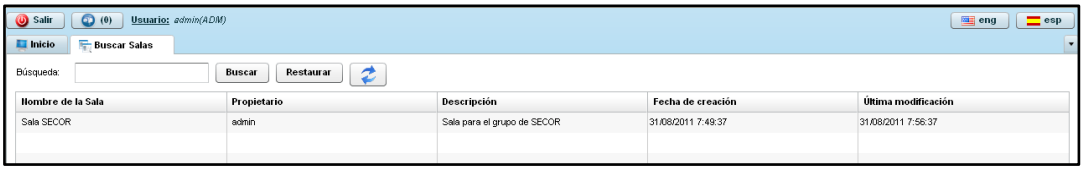


Figura 51: Diseño Final - Buscar Salas

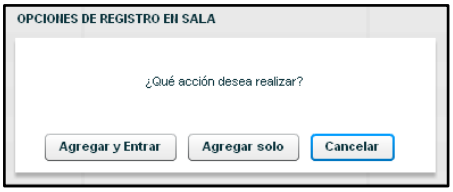


Figura 52: Diseño Final - Opciones al registrarse en la sala

Otra de las mejoras implementadas ha sido la inclusión de las invitaciones a salas. El usuario propietario o moderador de una sala, podrá enviar invitaciones a otros usuarios para que estos entren a participar dentro de la sala.

Figura 53: Diseño Final - Menú de gestión de usuarios de salas

El usuario que envía la invitación podrá seleccionar con qué función desea que el usuario destinatario participe dentro de la sala, tal y como se ve en la siguiente figura:

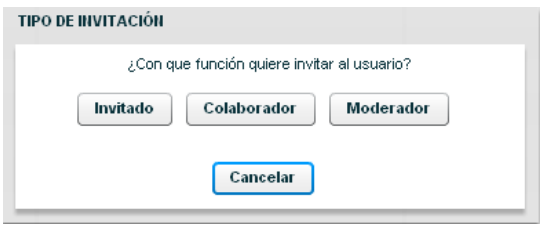


Figura 54: Diseño Final – Invitaciones

Por otro lado, el usuario destinatario de la invitación podrá visualizar las invitaciones que le han enviado en todo momento en la parte superior de la ventana:

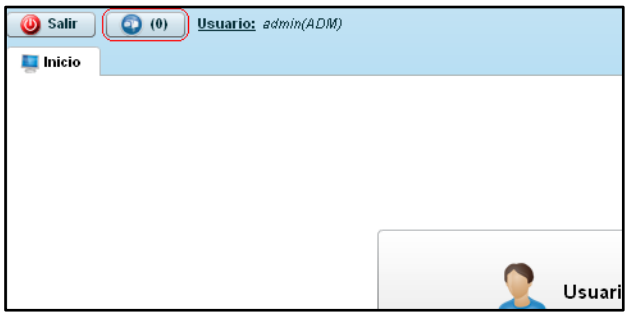


Figura 55: Diseño Final - Buzón de Invitaciones

Si el usuario está conectado cuando recibe una invitación, alado de su buzón de invitaciones aparecerá una notificación de que ha recibido una invitación nueva:

Figura 56: Diseño Final - Notificación de nueva invitación

El usuario podrá acceder mediante este botón (Figura X) a ver las invitaciones que ha recibido y aceptar () o declinar () dichas invitaciones:

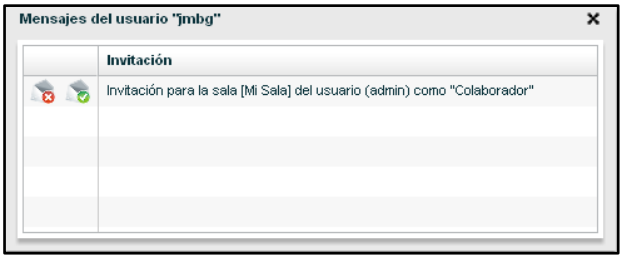


Figura 57: Diseño Final - Invitaciones de usuarios

Por último, la ventana de las salas se ha remodelado para adaptarlo al nuevo diseño general del sistema, y cambiando un poco la distribución de los distintos elementos que forman parte de la sala.

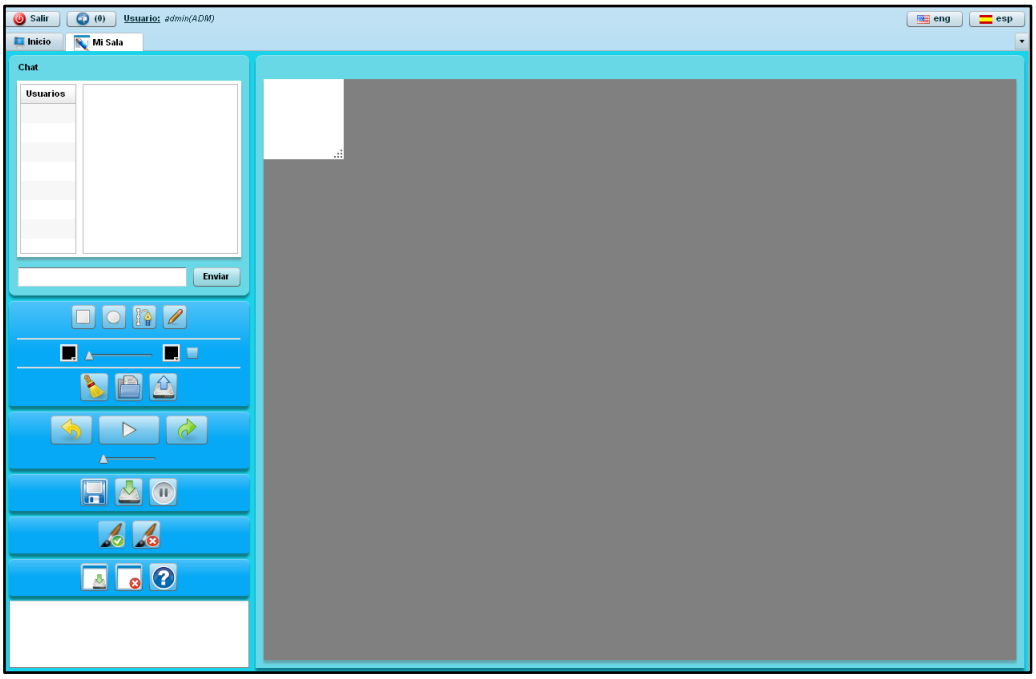


Figura 58: Diseño Final - Ventana de Sala

De forma adicional, se ha introducido un pequeño *textbox* en el cual se va reflejando un log de las distintas acciones que se van registrando en la sala a lo largo del tiempo.

De nuevo, dependiendo de la función que el usuario tenga dentro de la sala, podrá visualizar un conjunto de botones u otro.

Figura 59: Diseño Final - Acciones según función del usuario

4.2.7. Implementación

En este apartado de la memoria hablaremos sobre las distintas herramientas usadas para la implementación de la aplicación, la organización del proyecto y las peculiaridades del mismo.

Eclipse

Eclipse **[ECLP]** es un IDE (entorno integrado de desarrollo) multiplataforma libre para crear aplicaciones clientes de cualquier tipo que proporciona un editor de código sencillo, una compilación rápida y un entorno amigable (es sencillo de utilizar).

En sus orígenes, Eclipse fue creado por IBM, pero en la actualidad lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

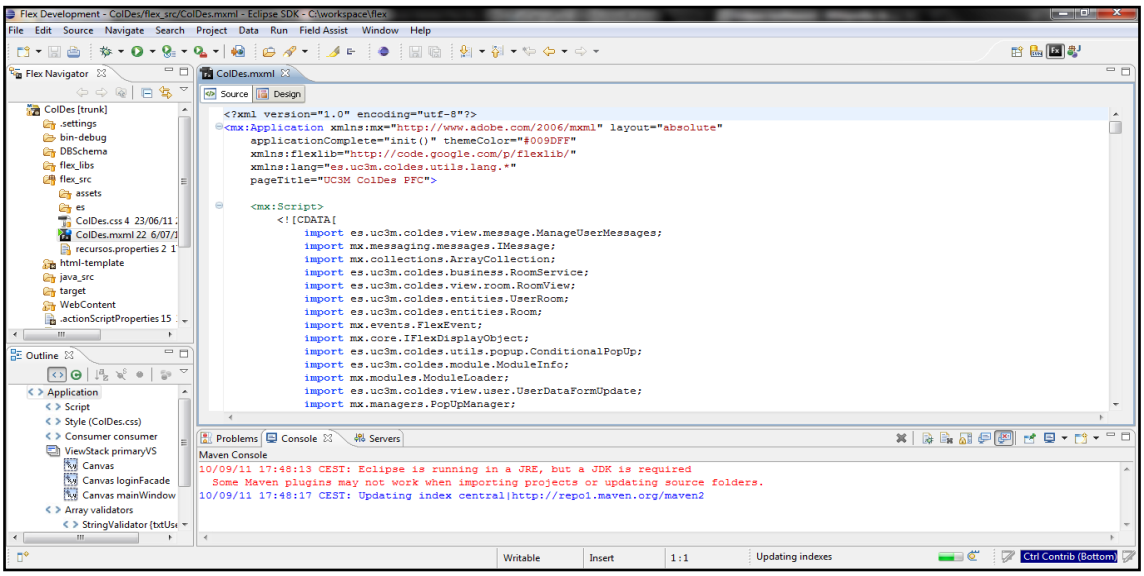


Figura 60: Eclipse IDE

Pese a que Eclipse esté escrito en su mayor parte en Java (salvo el núcleo), se ejecute sobre máquina virtual de ésta y su uso más popular sea como un IDE para Java, Eclipse es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc.

Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Un plug-in es un desarrollo aparte de la plataforma de Eclipse que aporta una determinada funcionalidad. de esta forma, se obtiene un entorno desarrollo totalmente extensible.

Uno de los plug-in desarrollados para este entorno es Adobe Flex Plug-in, mediante el cual se puede realizar desarrollos Flash en Eclipse. Sobre este plug-in hablaremos en el siguiente punto.

Este entorno de desarrollo tiene distintas versiones, de las cuales la usada para llevar a cabo la implementación de la aplicación ha sido la versión Ganymede, versión compatible con el plug-in mencionado anteriormente.

Actualmente, con las nuevas versiones de eclipse, Adobe ha lanzado un nuevo plug-in para adaptarse a estas nuevas versiones, donde entra en vigor también el cambio de nombre de Adode Flex a Adobe Flash: *Adobe Flash Builder 4 Plug-in*.

Adobe Flex 3 Plug-in

Adobe Flex 3 Plug-in **[FXPL]**, es un complemento para Eclipse con el que el desarrollador puede realizar proyectos para plataforma Flash. Es una adaptación del entorno Flash Builder para Eclipse.

Con este complemento, se permite al desarrollador crear código *Flash* y *ActionScript* para aplicaciones Web (*Flash*) y de escritorio (*AIR*).

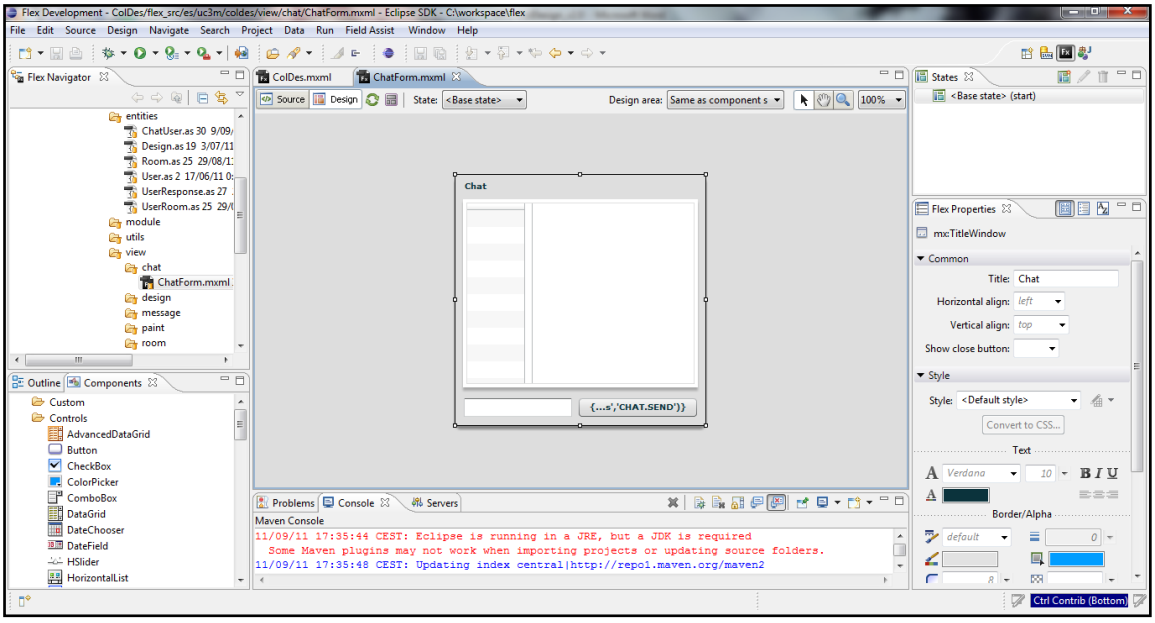


Figura 61: Eclipse + Adobe Flex Plug-in

BlazeDS

BlazeDS **[**BLDS**]** es un framework OpenSource (versión libre de LiveCycle) desarrollado por Adobe que permite a los desarrolladores comunicar fácilmente una aplicación back-end distribuida con una aplicación front-end desarrollada en Flex.

BlazeDS proporciona un sistema de servicios que permiten que clientes (Flex) y servidor se puedan comunicar en tiempo real, permitiendo a la aplicación Flex invocar métodos de objetos Java en el servidor.

Una aplicación basada en BlazeDS consta de 2 partes: Una cliente (Flex) y una aplicación web J2EE.

La arquitectura desde el lado cliente es la siguiente:

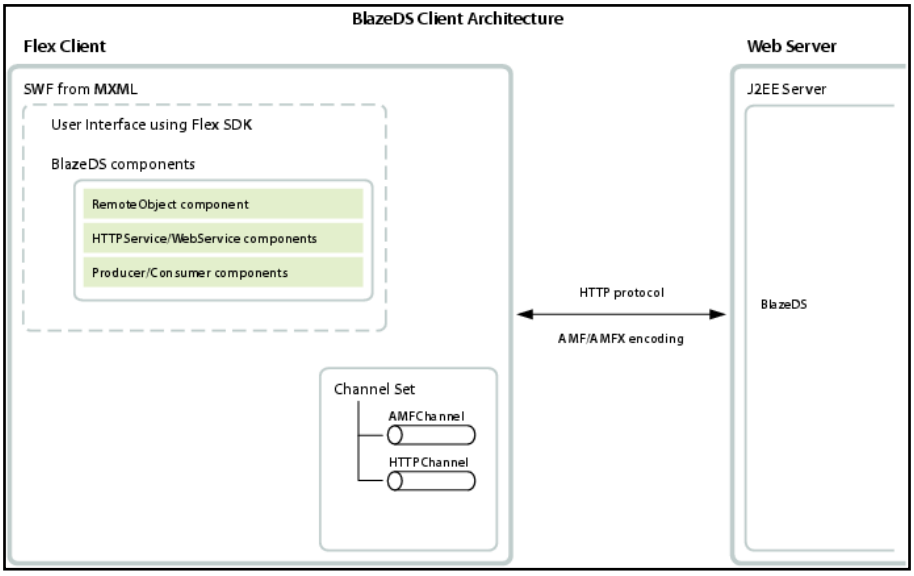


Figura 62: Parte Cliente BlazeDS

La arquitectura desde el lado servidor es la siguiente:

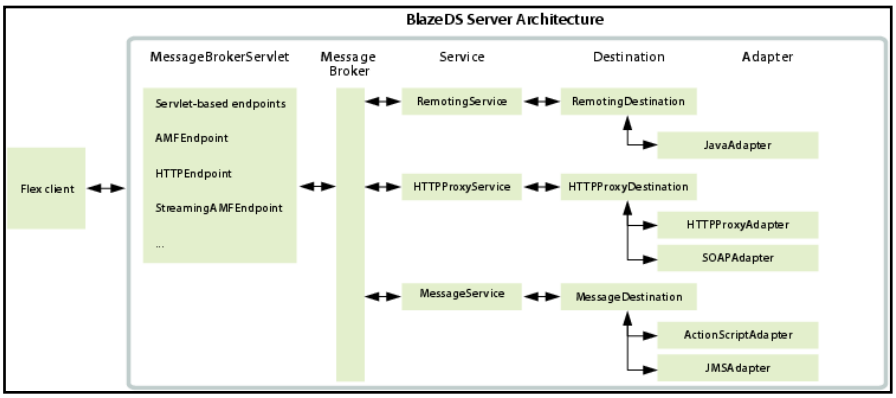


Figura 63: Parte Servidor BlazeDS

Apache Tomcat

Apache Tomcat **[TOMC]** es un contenedor de servlets, no un servidor de aplicaciones como JBOSS. Los servidores de aplicaciones son capaces de gestionar la mayor parte de las funciones de lógica de negocio y de acceso a los datos de una aplicación

A pesar del nombre (Apache Tomcat) no depende del servidor web Apache para su funcionamiento; Apache es una implementación en C de un servidor web HTTP que es capaz deservir paginas HTML.

Aunque en sus inicios existió la percepción de que el uso de Apache Tomcat de forma autónoma sólo era recomendable para entornos de desarrollo la realidad a día de hoy es que Apache Tomcat se puede usar como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Para nuestro desarrollo no se ha empleado la versión original de Apache Foundation, sino una alternativa Open Source proporcionada por Adobe Systems, incluida en el BlazeDS que incluye:

- Contenedor de servlets Java basado en Apache Tomcat 6, pero optimizado para el funcionamiento con Adobe Flex.
- Interfaces para conectar la parte servidora, desarrollada en Java, con la parte cliente, desarrollada en Flex.

Maven

La gestión y construcción de la parte Java del proyecto se ha agilizado mediante el uso de la herramienta Maven. Esta herramienta es similar un funcionamiento a Apache Ant, pero tiene un modelo de configuración mucho más simple basado en XML.

Maven **[MAVN]** usa un pom.xml (*Proyect Object Model*) para describir el proyecto software, las dependencias de otros módulos y componentes y el orden de construcción de los elementos. Con esta herramientas también podemos definir las tareas más habituales que se realizan con este tipo de proyectos, como compilado, generar .jar, .war, .ear, documentación, distribuir, dependencias con otros .jar, etc.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>prueba.mvn</groupId>
  <artifactId>primerProyecto</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>primerProyecto</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>activation</groupId>
      <artifactId>activation</artifactId>
      <version>1.0.2</version>
      <scope>compile</scope>
    </dependency>
  </dependencies>
</project>
```

Figura 64: Ejemplo de pom.xml

El punto fuerte de Maven es que el motor de su sistema puede hacer de forma dinámica la descarga de distintos elementos de un repositorio, desde plugins hasta componentes de otras organizaciones (como conectores a base de datos por ejemplo).

Organización del proyecto

El proyecto se ha desarrollado integro con el *IDE Eclipse* y con las herramientas de administración de *MySQL Server*. La organización de los distintos elementos del sistema se distribuye de la siguiente manera:

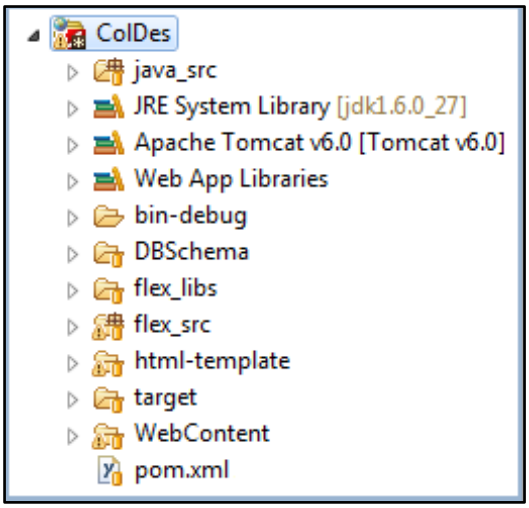
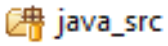


Figura 65: Esquema de carpetas del proyecto

En la carpeta 

está el código de la parte servidora de la aplicación, a continuación se muestran los distintos paquetes y clases que forman esta parte:

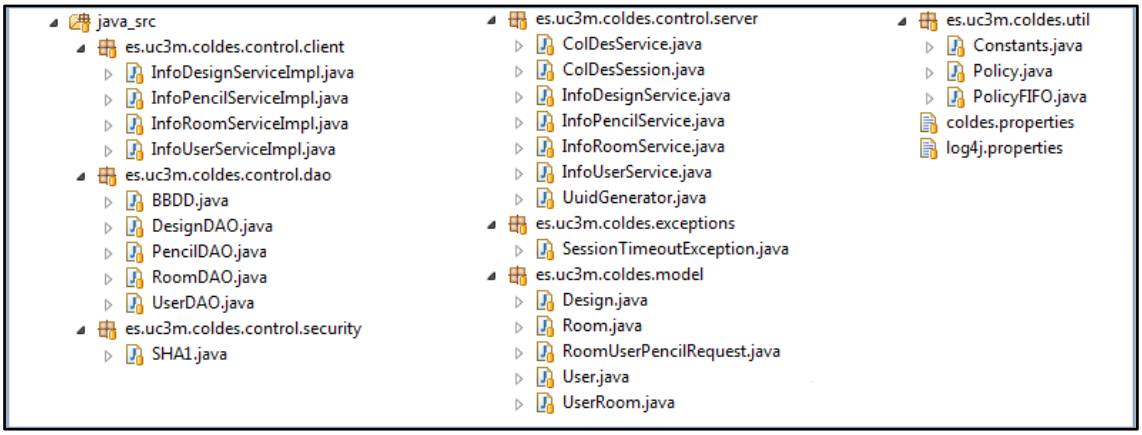


Figura 66: Esquema de clases java del proyecto

En la carpeta flex_src está el código de la parte cliente de la aplicación, a continuación se muestran los distintos paquetes y clases que forman esta parte:

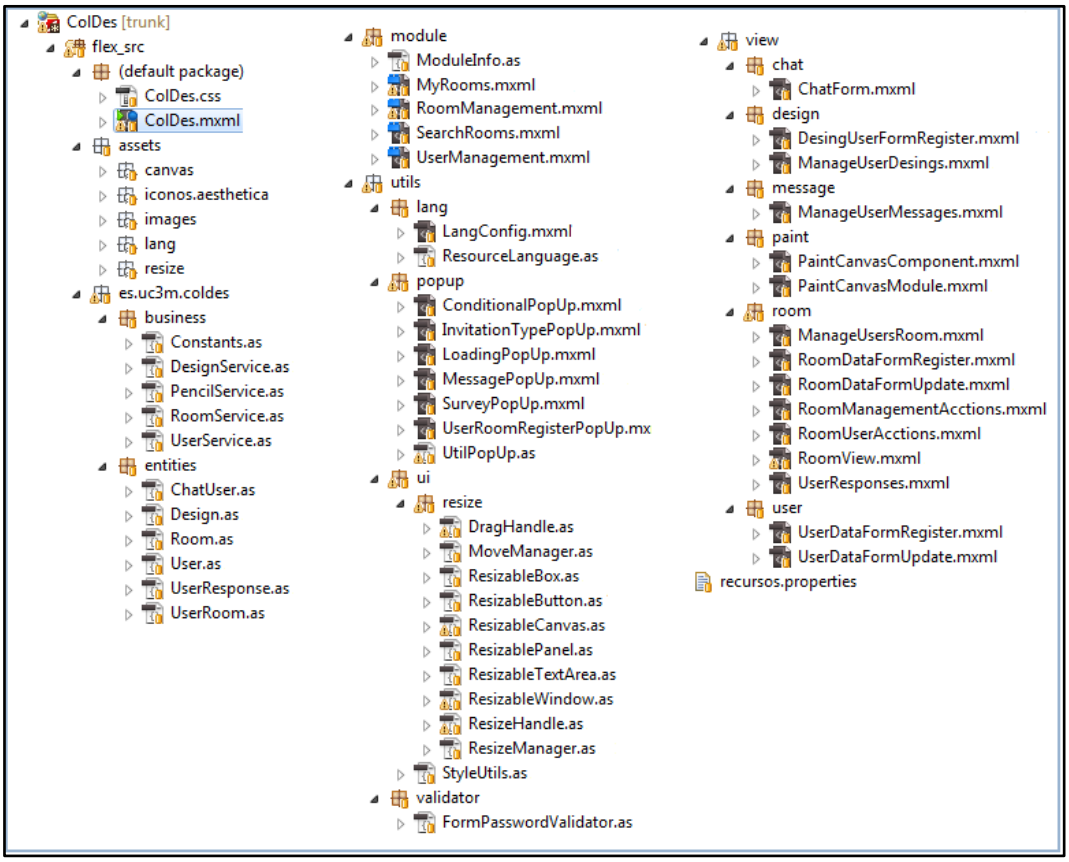


Figura 67: Esquema de componentes de la vista

5. Evaluación

La construcción de cualquier herramienta software tiene como objetivo satisfacer unas necesidades planteadas. En este apartado se evalúa si el sistema desarrollado satisface las necesidades presentadas en el apartado de objetivos de la introducción de este documento mediante una serie de pruebas que lo verifiquen.

5.1. Proceso de evaluación

En términos generales, se pueden distinguir dos tipos de evaluaciones durante el proceso de desarrollo: verificaciones y validaciones, donde cada tipo lo que busca es lo siguiente:

- **Verificación:** Determinada si el producto de la fase de desarrollo cumple o no los requisitos establecidos en la fase de análisis.
- **Validación:** Este proceso lo que busca es asegurar que el producto final del proceso de desarrollo cumple las necesidades del cliente.

La verificación tiene que ver típicamente con errores de los desarrolladores que no han construido bien el producto. Mientras que la validación tiene que ver con errores de los desarrolladores al malinterpretar las necesidades del cliente y por tanto no están construyendo correctamente la funcionalidad requerida.

En los siguientes apartados evaluaremos el producto desarrollado de las dos formas descritas: la verificación la realizaremos en base al plan de pruebas definido en el apartado de pruebas de este documento, y la validación se realizara poniendo a prueba el sistema frente a los usuarios finales del mismo.

5.1.1. Plan de pruebas

Para el plan de pruebas se han establecido una serie de pruebas que permitan verificar que el sistema cumple los requisitos básicos de calidad establecidos.

En este apartando se expondrá el plan de pruebas que se realizaran para verificar el correcto funcionamiento del sistema. Cada una de estas pruebas estará compuesta por los siguientes campos:

- **Identificador**: Numero de prueba.
- **Caso de uso**: Caso de uso asociado a la prueba que se realiza.
- **Descripción**: información sobre la prueba a realizar y que pretende validar.

Identificador	Caso de Uso	Descripción
PR-001	*CU-001*	Se prueba a acceder a la aplicación mediante un explorador que tiene instalado el plug-in de Adobe Flash Player y se verifica que la web se carga correctamente.
PR-002	*CU-001*	Se prueba a acceder a la aplicación mediante un explorador que no tiene instalador el plug-in de Adobe Flash Player y se comprueba que el explorador solicita al usuario descargar el plug-in para una correcta visualización de la web.
PR-003	*CU-002*	Se verifica que al intentar acceder con unas credenciales (usuario y contraseña) correctas el sistema autoriza al usuario y se carga la página principal de la web.
PR-004	*CU-002*	Se intenta acceder a la aplicación con unas credenciales incorrectas y se comprueba que el sistema advierte al usuario que el usuario con el que se intenta acceder es incorrecto. Tras esto se verifica que se puede volver a intentar el acceso a la aplicación.
PR-005	*CU-003*	Se verifica que los campos al crear un nuevo usuario son: nombre, apellidos, correo, nombre de usuario y contraseña.
PR-006	*CU-003*	Se prueba a registrar un usuario introduciendo todos los datos correctamente y se verifica que el registro se realiza con éxito.
PR-007	*CU-003*	Se prueba a registrar un usuario introduciendo algún dato incorrecto y se verifica que el sistema notifica al usuario los campos erróneos del registro.
PR-008	*CU-003*	Se prueba a registrar un usuario introduciendo todos los datos correctamente pero usando un nombre de usuario ya existente y se verifica que el sistema notifica al usuario que el nombre de usuario que está usando para llevar a cabo el registro ya existe.
PR-009	*CU-004*	Se prueba a autenticarse en el sistema con un usuario correcto y activo y se verifica que se accede correctamente al sistema.
PR-011	*CU-004*	Se prueba a autenticarse en el sistema con un usuario correcto pero no activo y se verifica que el sistema notifica al usuario que el usuario no es correcto.
PR-010	*CU-004*	Se prueba a autenticarse en el sistema con un usuario incorrecto y se verifica el sistema notifica al usuario que alguno de los campos introducidos no es correcto (usuario o contraseña).
PR-011	*CU-005*	Se verifica que un usuario administrador puede acceder al modulo de gestión de usuarios y visualizar todos los usuarios registrados en el sistema.
PR-012	*CU-006*	Se verifica que un usuario administrador puede crear nuevos usuarios desde el modulo de gestión de usuarios del sistema.
PR-013	*CU-006*	Se verifica que los campos al crear un nuevo usuario desde el modulo de gestión de usuarios son: nombre, apellidos, correo, nombre de usuario, contraseña, perfiles dentro del sistema y si esta activo o no.
PR-014	*CU-006*	Se prueba a registrar desde el modulo de gestión de usuarios un usuario introduciendo todos los datos correctamente y se verifica que el registro se realiza con éxito.
PR-015	*CU-006*	Se prueba a registrar desde el modulo de gestión de usuarios un usuario introduciendo algún dato incorrecto y se verifica que el sistema notifica al usuario los campos erróneos del

Identificador	Caso de Uso	Descripción
		registro.
PR-016	*CU-007*	Comprobamos que un usuario administrador puede modificar los datos de cualquier usuario registrado en el sistema desde el modulo de gestión de usuarios.
PR-017	*CU-007*	Se verifica que los campos que el usuario administrador puede modificar de cualquier usuario desde el modulo de gestión de usuarios son: nombre, apellidos, correo, contraseña, perfiles dentro del sistema y si esta activo o no.
PR-018	*CU-007*	Se comprueba que al cancelar la actualización de los datos de un usuario desde el modulo de gestión de usuarios estos no se modifican.
PR-018	*CU-007*	Se comprueba que al llevar a cabo la actualización de los datos de un usuario desde el modulo de gestión de usuarios estos se modifican correctamente.
PR-019	*CU-008*	Se verifica que un usuario administrador puede acceder al modulo de gestión de salas y visualizar todos las salas registradas en el sistema.
PR-020	*CU-009*	Comprobamos que un usuario administrador puede modificar los datos de cualquier sala registrada en el sistema desde el modulo de gestión de salas.
PR-021	*CU-009*	Se verifica que los campos que el usuario administrador puede modificar de cualquier sala desde el modulo de gestión de salas son: nombre de la sala, privacidad de la sala, estado de la sala, tipo de participación en la sala y descripción de la misma.
PR-022	*CU-009*	Se comprueba que al cancelar la actualización de los datos de una sala desde el modulo de gestión de salas estos no se modifican.
PR-023	*CU-009*	Se comprueba que al llevar a cabo la actualización de los datos de una sala desde el modulo de gestión de salas estos se modifican correctamente.
PR-024	*CU-010*	Se verifica que un usuario administrador desde el módulo de gestión de salas puede gestionar los usuarios participantes de una sala, viendo los actuales participantes de la sala, y el resto de usuarios no participantes en la misma.
PR-025	*CU-011*	Se verifica que el usuario administrador puede enviar invitaciones desde el modulo de gestión de salas a cualquier usuario no participante de una sala, escogiendo en el momento en el que se envía la invitación la función que este va a desempeñar en la sala.
PR-026	*CU-012*	Se verifica que un usuario administrador puede expulsar usuarios participantes de una sala desde el modulo de gestión de salas.
PR-027	*CU-013*	Se comprueba que un usuario administrador desde el modulo de gestión de salas puede modificar la función de cualquier usuario dentro de una sala.
PR-028	*CU-013*	Se verifica que cuando un usuario administrador desde el módulo de gestión de salas modifica la función de un usuario dentro de una sala y actualiza la sala, lo cambios se guardan correctamente.
PR-029	*CU-013*	Se verifica que cuando un usuario administrador desde el

Identificador	Caso de Uso	Descripción
		módulo de gestión de salas modifica la función de un usuario dentro de una sala pero cancela el cambio, lo cambios se no se modifican.
PR-030	*CU-014*	Se verifica que un usuario diseñador puede acceder a sus datos desde la ventana principal del sistema: nombre, apellidos, correo electrónico, nombre de usuario y contraseña.
PR-031	*CU-015*	Se verifica que un usuario diseñador puede modificar a sus datos desde la ventana principal del sistema: nombre, apellidos, correo electrónico y contraseña.
PR-032	*CU-016*	Se verifica que un usuario diseñador y administrador puede acceder a sus datos desde la ventana principal del sistema: nombre, apellidos, correo electrónico, nombre de usuario, contraseña, perfiles del usuario y si esta activo o no.
PR-033	*CU-017*	Se verifica que un usuario diseñador y administrador puede modificar a sus datos desde la ventana principal del sistema: nombre, apellidos, correo electrónico, contraseña, perfiles del usuario y si esta activo o no.
PR-034	*CU-018*	Se comprueba que un usuario diseñador puede acceder al menú de salas personales desde la ventana principal y ver todas las salas en las cuales está participando el usuario.
PR-035	*CU-019*	Se verifica que los campos al crear una nueva sala son desde el la ventana de salas personales son: nombre de la sala, privacidad de la sala, tipo de participación en la sala y descripción de la misma.
PR-036	*CU-019*	Se prueba a crear una nueva sala son desde el la ventana de salas personales introduciendo todos los datos correctamente y se verifica que el registro se realiza con éxito.
PR-037	*CU-019*	Se prueba a crear una nueva sala son desde el la ventana de salas personales introduciendo algún dato incorrecto y se verifica que el sistema notifica al usuario los campos erróneos del registro.
PR-038	*CU-019*	Se prueba a crear una nueva sala son desde el la ventana de salas personales introduciendo como nombre de la sala un nombre de sala ya existente para el usuario que está creando la sala, y se verifica que el sistema notifica al usuario que ya tiene una sala con ese nombre.
PR-039	*CU-020*	Se verifica que cualquier usuario con función dentro de la sala de moderador, colaborador o invitado puede dejar de participar en una sala.
PR-040	*CU-020*	Se comprueba que si un usuario con función dentro de la sala de moderador, colaborador o invitado acepta dejar de participar en una sala, esta deja de aparecer entre sus salas personales.
PR-041	*CU-020*	Se comprueba que si un usuario con función dentro de la sala de moderador, colaborador o invitado cancela dejar de participar en una sala, esta deja sigue apareciendo entre sus salas personales.
PR-042	*CU-021*	Se verifica que cualquier usuario con función dentro de la sala de propietario puede dejar de participar en una sala.
PR-043	*CU-021*	Se comprueba que si un usuario con función dentro de la sala de propietario acepta dejar de participar en una sala, esta deja

Identificador	Caso de Uso	Descripción
		de aparecer entre sus salas personales y se borra del sistema.
PR-044	*CU-021*	Se comprueba que si un usuario con función dentro de la sala de propietario cancela dejar de participar en una sala, esta deja sigue apareciendo entre sus salas personales y no se borra del sistema.
PR-045	*CU-021*	Se verifica que a la hora de borrar una sala, se eliminan también todos los elementos relacionados con la misma: relaciones con usuarios, con diseños y con invitaciones pendientes.
PR-046	*CU-022*	Se comprueba que únicamente el usuario propietario de una sala puede modificar los datos relativos a misma desde la ventana de sala personales del usuario.
PR-047	*CU-022*	Se verifica que los campos que el usuario propietario puede modificar de las salas donde desempeña esa función son: nombre de la sala, privacidad de la sala, estado de la sala, tipo de participación en la sala y descripción de la misma.
PR-048	*CU-022*	Se comprueba que al cancelar la actualización de modificación de los datos desde la ventana de salas personales estos no se modifican.
PR-049	*CU-022*	Se comprueba que al realizar la actualización de modificación de los datos desde la ventana de salas personales estos se modifican correctamente.
PR-050	*CU-023*	Se comprueba que únicamente los usuarios propietarios y moderadores de una sala gestionar los usuarios participantes de la misma, viendo los actuales usuarios participantes, y el resto de usuarios no participantes desde la ventana de salas personales.
PR-051	*CU-024*	Se verifica que únicamente los usuarios propietarios y moderadores de una sala pueden enviar invitaciones desde la ventana de salas personales a cualquier usuario no participante de una sala, escogiendo en el momento en el que se envía la invitación la función que este va a desempeñar en la sala.
PR-052	*CU-025*	Se verifica que únicamente los usuarios propietarios y moderadores pueden expulsar usuarios participantes de una sala desde la ventana de salas personales.
PR-053	*CU-026*	Se comprueba que únicamente los usuarios propietarios y moderadores pueden modificar la función de cualquier usuario dentro de una sala desde la ventana de salas personales.
PR-054	*CU-026*	Se verifica que cuando un usuario propietario o moderador modifica la función de un usuario dentro de una sala y actualiza la sala, lo cambios se guardan correctamente.
PR-055	*CU-026*	Se verifica que cuando un usuario propietario o moderador modifica la función de un usuario dentro de una sala pero cancela el cambio, lo cambios se no se modifican.
PR-056	*CU-027*	Se prueba a acceder desde la ventana de salas personales a cualquier sala en la que este participando el usuario y se verifica que el sistema carga correctamente la sala.
PR-057	*CU-028*	Se comprueba que un usuario diseñador puede acceder al menú de búsqueda de salas del sistema desde la ventana principal y ver únicamente las salas públicas registradas en el sistema.

Identificador	Caso de Uso	Descripción
PR-058	*CU-029*	Verificamos que cualquier usuario puede apuntarse a participar en cualquier sala pública del sistema desde la ventana de búsqueda de salas y que la función que este desempeñara en la sala será de colaborador.
PR-059	*CU-030*	Se verifica que cualquier usuario pude apuntarse y acceder de forma directa a cualquier sala pública del sistema desde la ventana de búsqueda de salas y que la función que este desempeñara en la sala será de colaborador.
PR-060	*CU-031*	Se comprueba que un usuario cuya función sea propietario, moderador o colaborador puede pintar en el lienzo de una sala cuyo tipo de participación es de uno en uno si tiene el pincel.
PR-061	*CU-031*	Se comprueba que un usuario cuya función sea propietario, moderador o colaborador no puede pintar en el lienzo de una sala cuyo tipo de participación es de uno en uno si no tiene el pincel.
PR-062	*CU-032*	Se prueba que un usuario sea propietario, moderador o colaborador puede pintar en cualquier momento sobre el lienzo de una sala si el tipo de participación de esta es de todos a la vez.
PR-063	*CU-033*	Se verifica que un usuario cuya función sea propietario, moderador o colaborador puede solicitar el pincel en cualquier momento siempre y cuando no tenga el pincel ya.
PR-064	*CU-033*	Se comprueba que un usuario al recibir el pincel puede comenzar a dibujar sobre el lienzo de una sala, y puede dejar el pincel.
PR-063	*CU-034*	Se verifica que un usuario cuya función sea propietario, moderador o colaborador puede dejar el pincel en cualquier momento siempre y cuando lo tenga.
PR-064	*CU-034*	Se comprueba que un usuario al dejar el pincel ya no puede dibujar sobre el lienzo de una sala, y puede volver a solicitar el pincel.
PR-065	*CU-035*	Se prueba que un usuario cuya función sea propietario, moderador o colaborador puede exportar a una imagen local el contenido del lienzo de la sala especificando la ruta del pc en la cual quiere almacenar la imagen en cualquier momento.
PR-066	*CU-036*	Se prueba que un usuario cuya función sea propietario, moderador o colaborador puede importar a una imagen local al contenido del lienzo de la sala especificando la ruta del pc en la cual se encuentra la imagen únicamente cuando el usuario puede realizar acciones sobre el lienzo de la sala.
PR-067	*CU-037*	Se verifica que un usuario cuya función sea propietario, moderador o colaborador puede guardar el contenido del lienzo de una sala como un diseño propio.
PR-068	*CU-038*	Se comprueba que un usuario cuya función sea propietario, moderador o colaborador puede cargar al lienzo de la sala un diseño guardado anteriormente, siempre y cuando pueda realizar acciones sobre el lienzo de la sala.
PR-069	*CU-039*	Se comprueba que cualquier usuario participante de una sala puede comunicarse con el resto de usuarios a través del chat de la sala.
PR-070	*CU-040*	Se prueba que un usuario cuya función sea propietario o

Identificador	Caso de Uso	Descripción
		moderador puede asignar el contenido del lienzo a la sala.
PR-071	*CU-040*	Se verifica que si hay un diseño asignado a una sala, en el momento que un usuario entra a la sala este es el diseño que aparece en el lienzo por defecto.
PR-072	*CU-041*	Se prueba que un usuario cuya función sea propietario o moderador puede eliminar un diseño del contenido del lienzo de la sala.
PR-073	*CU-041*	Se verifica que si no hay un diseño asignado a una sala, en el momento que un usuario entra a la sala el diseño que aparece en el lienzo estará vacio.
PR-074	*CU-042*	Se verifica que cualquier usuario puede ver las distintas acciones realizadas sobre el lienzo mediante los botones de hacer y rehacer.
PR-075	*CU-043*	Se comprueba que un usuario diseñador es capaz de visualizar en cualquier momento las invitaciones que este tiene para participar en las distintas salas del sistema.
PR-076	*CU-043*	El usuario desde la ventana de invitaciones puede aceptar o rechazar invitaciones a participar en salas, tras realizar cualquier acción la invitación desaparece.
PR-077	*CU-043*	Cuando un usuario acepta participar en una sala mediante una invitación este podrá acceder a la sala de forma directa desde la ventana de salas personales.
PR-078	*CU-043*	Cuando un usuario rechaza participar en una sala, se verifica que únicamente se borra la invitación.

Tabla 72: Catalogo de Pruebas

En la siguiente tabla se presentan los resultados obtenidos de cada una de las pruebas definidas en la tabla anterior:

Prueba	001	002	003	004	005	006	007	008	009	010
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	011	012	013	014	015	016	017	018	019	020
Resultado	Éxito	Fallo	Fallo	Fallo	Fallo	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	021	022	023	024	025	026	027	028	029	030
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	031	032	033	034	035	036	037	038	039	040
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	041	042	043	044	045	046	047	048	049	050
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	051	052	053	054	055	056	057	058	059	060
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	061	062	063	064	065	066	067	068	069	070
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
Prueba	071	072	073	074	075	076	077	078		
Resultado	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito		

Tabla 73: Resultados del Plan de Pruebas I

Los fallos localizados en las pruebas únicamente hacen referencia a la creación de usuarios desde el modulo de gestión de usuarios por parte de los administradores. A la hora de realizar las pruebas se verifico que no se daba la opción en ningún lugar de la pantalla de gestión para la creación de nuevos usuarios.

Tras una nueva implementación de este modulo se verifico que un usuario administrador podía crear nuevos usuarios desde esta ventana.

Prueba	012	013	014	015
Resultado	Éxito	Éxito	Éxito	Éxito

Tabla 74: Resultados del Plan de Pruebas II

5.1.2. Evaluación de los usuarios

Para obtener una validación por parte de los usuarios finales del sistema, la aplicación ha sido desplegada en los entornos de trabajo de los usuarios para que estos pudiesen probar más a fondo la aplicación y validar así si el sistema cumple las necesidades que estos presentaban.

Tras analizar los resultados se verifica que las funcionalidades básicas definidas en los objetivos de este documento se cumplen. No obstante, tras las pruebas realizadas por los usuarios se han detectado una serie de errores (incidencias) y peticiones de los usuarios de mejora de la aplicación. El documento facilitado por los usuarios con estas incidencias se adjunta con el documento en el Anexo VII.

De forma adicional, para complementar estas carencias que los usuarios encontraron de la aplicación, se ha dividido el documento facilitado por los mismos en distintas incidencias y mejoras, que posteriormente fueron implementadas para satisfacer a los usuarios. Toda esta información se recoge en el Anexo VIII (*Pruebas de Aceptación del Usuario*).

6. Conclusión

Como punto final a este documento, en este último apartado se presentan las conclusiones obtenidas por el autor a lo largo del desarrollo del proyecto. En él se analizan las aportaciones realizadas, las posibles líneas de mejora y ampliación, y las dificultades encontradas a lo largo del proyecto.

6.1. Aportaciones realizadas

La consecución de este proyecto ha derivado en la generación de una herramienta mediante la cual los usuarios pueden colaborar de forma distribuida con otros usuarios mediante el uso de pizarras online en las diversas salas creadas y gestionadas por ellos mismos dentro de la aplicación.

Se establece de esta forma un entorno de trabajo distribuido para el trabajo colaborativo entre personas que se encuentran en lugares y contextos distintos, mediante el cual pueden poner en común las ideas que puedan tener y plasmarlas en un diseño grafico mediante el uso de las pizarras online.

Se incluye además la posibilidad de participar en las salas de dos formas distintas, aportando de este modo a los usuarios un entorno donde, según las necesidades de colaboración de un determinado proyecto, pueden crear salas que se adapten de mejor forma a estas necesidades alternando participación por turnos y participación de todos a la vez.

6.2. Trabajos futuros

En esta sección se explican las diversas posibilidades que se contemplan para la continuación del proyecto explicado en este documento. La mayor parte de los trabajos futuros contemplados en este apartado son mejoras que afectarían muy positivamente a la aplicación.

Una de las mejoras consideradas para el proyecto es el cambio de la forma de almacenamiento de los diseños en el sistema. Actualmente los diseños se almacenan de forma directa en base de datos como un array de bytes que representa el contenido del mismo. Esta forma de almacenamiento tiene un problema que radica en el tamaño de las imágenes que estén contenidas en el lienzo. Los campos definidos en base de datos para almacenar este valor tienen el tipo de *text* o *longtext* y tienen un valor máximo, por lo que en el momento que el contenido del lienzo tenga un tamaño mayor que el máximo de este tipo, el sistema fallara al guardar el diseño.

Por ello se considera una interesante línea futura cambiar el modo de almacenamiento de forma que lo que se haga realmente es almacenar el contenido del lienzo como una imagen en el servidor del sistema, y desde base de datos hacer referencia a esta imagen en el servidor, para ello sería necesario definir unos servlets encargados de la subida de la imagen al servidor y de la descarga posterior al lienzo a la hora de cargarlo en una sala.

Por otro lado, del lado de la colaboración en salas, en la actual aplicación una sala únicamente puede tener un lienzo sobre el cual pueden estar trabajando una o más personas. En este punto sería interesante modificar una sala aceptara más de un lienzo, de tal forma que en una misma sala se pudiese estar trabajando sobre distintos diseños sin necesidad de tener varias salas abiertas.

De forma adicional, a la hora de trabajar dentro del lienzo de la sala, una de las mejoras que se podrían realizar del proyecto es el tratamiento específico de cada una de las figuras dibujadas sobre el lienzo, pudiendo cambiar cualquier característica de estas una vez que hayan sido dibujadas.

Por último, sería interesante, en caso de verse necesario, mejorar la usabilidad de la herramienta. La usabilidad web se refiere a: "*la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso*" [ISO/IEC 9126]. Los principios básicos de usabilidad que ha de seguir cualquier web son los siguientes **[LWBR]**:

- **Facilidad de Aprendizaje:** facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema.
- **Flexibilidad:** relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información.
- **Robustez:** es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos.

Para ello sería necesario detectar las carencias de la herramienta en cuanto a usabilidad se refiere y en base a los resultados modificar la interfaz, e incluso funcionalidad, para cumplir estos principios.

6.3. Problemas encontrados

Aunque el resultado global del proyecto es positivo y se han cumplido los objetivos establecidos, éste no ha estado exento de problemas y dificultades.

La principal dificultad a la que me he enfrentando a lo largo del proyecto de fin de carrera era la de ponerse al frente de un proyecto de esta envergadura de forma individual, llevando a cabo todas las decisiones que este tipo de proyecto requiere: gestión de tiempos, análisis de necesidades, diseño de la aplicación, etc.

Otro punto problemático que surgió en el desarrollo del proyecto fue el diseño de la interfaz gráfica del sistema, pues se busco que fuese lo más intuible y simple posible, pero sin dejar de lado en ningún momento las funcionalidades que el sistema tenía que realizar. De ahí que el sistema haya pasado por tantos distintos tipos de interfaces a lo largo del desarrollo.

Por último, el problema más importante que he encontrado a lo largo del desarrollo de este proyecto ha sido el de coordinar y combinar vida laboral con la propia realización del proyecto de fin de carrera, motivo por el cual la elaboración del proyecto fin de carrera ha llevado más tiempo del que en principio debería de haber llevado. No

obstante, tal y como se puede observar en la planificación adjunta en el Anexo II, la desviación de tiempos tampoco ha sido tan pronunciada como cabía esperar, ya que se dio por hecho que el proyecto se realizaría aproximadamente en 12 meses y finalmente se ha realizado en poco más de 13.

6.4. Opiniones personales

Es importante remarcar para empezar que la consecución de este proyecto de fin de carrera es un hito muy importante dentro de mis 25 años de estudiante, un momento únicamente equiparable al instante en el que era conocedor de mi aprobado en la última asignatura de la carrera, momentos muy especiales que no se olvidan con facilidad.

Con este proyecto, como ya he comentado en el punto anterior, me he encontrado con dificultades a la hora de enfrentarme a un proyecto de estas características de forma individual, pero, no hay mal que por bien no venga, y esto me ha servidor para aprender a planificar de forma más eficiente mi tiempo, a tomar decisiones importante dentro de proyectos de este tipo y a ir superando, pasito a pasito, los distintos problemas que se han presentado a lo largo de este último año.

Ha sido complicado combinar estudios con trabajo, pero no obstante he sido capaz de sacarle provecho realizando un proyecto en una tecnología en la cual me encontraba inmerso ya en el trabajo, de forma que he podido profundizar mucho más en ella y sacarle el máximo partido posible dentro del proyecto fin de carrera que he realizado.

Queda la espina, que me sacare tras la presentación de este proyecto, de no haber podido estudiar mucho más a fondo una tecnología emergente como es HTML5. No obstante, la realización de este proyecto me ha permitido realizar un estudio de esta tecnología y de las enormes posibilidades que ofrece no a muy largo plazo.

Como última conclusión, destacaría que la realización de este proyecto me ha ayudado a ver lo que un proyecto conlleva, desde su nacimiento como idea, hasta su consecución final, en este caso, como aplicación web.

7. Bibliografía

En este apartado de la memoria se encuentran todas las referencias usadas para la elaboración de la misma:

[ACRO10] Equipo de Acrobat. *10 Cosas Geniales Que Puede Hacer Acrobat.com*. 2010.
Disponible mediante registro [Internet]:
<https://acrobat.com >
[25 de Enero de 2011]

[ADOBW] Web Adobe [Internet]:
<http://www.adobe.com/products/player_census/flashplayer/>
[25 de Enero de 2011]

[BDI] PDF de bdigital.eafit.edu [Internet]:
<http://bdigital.eafit.edu.co/bdigital/PROYECTO/P620.0042E821/marcoTeorico.pdf>
[24 de Enero de 2011]

[Bez+09] Bezzi, M., Duquenoy, P., Fischer, S,.Hansen, M., Zhang, G. *Privacy and Identity Management for Life*. 5th IfipWg 9.2, 9.6/11.4, 11.6, 11.7/Primelife International Summer School, Nice, France, Revised Selected Papers. Septiembre 2009.
Disponible [Internet]
<http://books.google.es/books?id=peEsGDHxPKwC>
[25 de Enero de 2011]

[BYOUT] Blog de Youtube [Internet]:
<http://apiblog.youtube.com/2010/06/flash-and-html5-tag.html>
[22 de Marzo de 2011]

[CYCLW] Web Cycle [Internet]:
<www.cycle-it.com/#/rias/empresas/>
[25 de Enero de 2011]

[DABBL] Daddleboard [Internet]:
<http://www.dabbleboard.com>
[25 de Enero de 2011]

[Dei+08] Paul J. Deitel, Harvey M. Deitel. *AJAX, Rich Internet Applications, and Web Development for Programmers*.Prentice Hall, 2008.
Disponible [Internet]
<http://books.google.es/books?id=i_mieZ8QXMAC >
[24 de Enero de 2011]

[Fad+03] Fadi P. Deek, James A. McHugh. *Computer-Supported Collaboration with Applications to Software Development*. Springer, 2003.
Disponible [Internet]
<http://books.google.es/books?id=-K2S4m0TbjMC&dq>
[01 de Marzo de 2011]

[FOCUS] focus.com [Internet]:
<http://www.focus.com/fyi/information-technology/wtf-is-html5/>
[18 de Marzo de 2011]

[Hay+10] Hayward P. Andres and Obasi H. Akan, University Greensboro. *Assessing Team Learning during Technology-Mediated Collaboration*.2010.
Disponible [Internet]:
<http://www.iiis.org/CDs2010/CD2010IMC/CCCT_2010/PapersPdf/TA159VG.pdf>
[24 de Enero de 2011]

[IBER10] Ibero.wiki.nmc.org [Internet]:
<http://ibero.wiki.nmc.org/2010+Short+List+Entornos+colaborativos>
[25 de Enero de 2011]

[JMM07] Blog jmmanrique [Internet]:
<http://jmmanrique.blogspot.com/2007/09/web-tradicional-vs-web-20-blogs-en.html>
[01 de Marzo de 2011]

[Lon95] London, Scott. *Collaboration and Community*.Report 1995.
Disponible [Internet]:
<http://www.scottlondon.com/reports/collaboration.pdf>
[24 de Enero de 2011]

[Mar03] Martínez, Francisco. *Redes de comunicación en la enseñanza*. Ediciones Paidós Ibérica. 2003.
Disponible [Internet]
<http://books.google.es/books?id=jO-BXLrxVdwC>
[25 de Enero de 2011]

[MARK] Riaction.es [Internet]:
<http://marketshare.hitslink.com/browser-market-share.aspx?qprid=1>
[10 de Marzo de 2011]

[Mur+01] S. Murugesan, Y. Deshande. *I Jornadas de Ingeniería Web de la Ingeniería Web como nueva disciplina*. 2001.

[Nas05] Nascimbene, Carlos. *¿Qué son las Rich Internet Applications?*. Artículo 16 de Diciembre de 2005.
Disponible [Internet]:
<http://www.canal-ar.com.ar/noticias/noticiamuestra.asp?Id=2639>
[24 de Enero de 2011]

[RAE01] Real Academia de la Lengua Española. *Diccionario de la Lengua Española*. Edición 22ª. 2001. ISBN: ISBN: 97-884-2396-8145.
Disponible [Internet]:
<http://www.rae.es>
[24 de Enero de 2011]

[RIACT] Riaction.es [Internet]:
<http://riaction.es/servicios.html>
[25 de Enero de 2011]

[She+08] Shen, W., Hao, Q., Li, W. *Computer supported collaborative design: retrospective and perspective*. Diciembre, 2008.
Disponible [Internet]:
<http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=5755851>
[01 de Marzo de 2011]

[COTSEG] Cotización Seguridad social.
Disponible [Internet]:
<http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm#>
[16 de Noviembre de 2011]

[MICPR] Precio Microsoft Project 2010.
Disponible [Internet]:
< http://emea.microsoftstore.com/es/es-ES/Microsoft/Project-Professional-2010>
[20 de Noviembre de 2011]

[MICOF] Precio Microsoft Office 2010.
Disponible [Internet]:
<http://emea.microsoftstore.com/es/es-ES/Microsoft/Office-Hogar-y-Pequena-Empresa-2010>
[20 de Noviembre de 2011]

[SERUB] Precio Servidor VPS Ubuntu 10.04 LTS de 1024MB de RAM:
Disponible [Internet]:
<http://alta.1and1.es/VirtualServerXL>
[20 de Noviembre de 2011]

[ECLP] The Eclipse Foundation open source community website:
Disponible [Internet]:
<http://www.eclipse.org/>
[27 de Noviembre de 2011]

[FXPL] Adobe Flex Builder 3.0.2 Professional Eclipse Plug-in:
Disponible [Internet]:
<https://www.adobe.com/cfusion/tdrc/index.cfm?product=flex_eclipse>
[27 de Noviembre de 2011]

[BLDS] BlazeDS Developer Guide:
Disponible [Internet]:
<http://livedocs.adobe.com/blazeds/1/blazeds_devguide/>
[27 de Noviembre de 2011]

[TOMC] Apache Tomcat website:
Disponible [Internet]:
<http://tomcat.apache.org/>
[27 de Noviembre de 2011]

[MAVN] Apache Maven Project:
Disponible [Internet]:
<http://maven.apache.org/>
[27 de Noviembre de 2011]

[LWBR] Descripción de usabilidad web (lawebera.es):
Disponible [Internet]:
<http://www.lawebera.es/de0/usabilidad.php>
[12 de Diciembre de 2011]

[OBOA] Descripción de usabilidad web (lawebera.es):
Disponible [Internet]:
<http://www.ehow.com/facts_5797055_online-whiteboard_.html>
[28 de Diciembre de 2011]

Anexo I. Control de versiones

Este anexo contiene las distintas versiones por las que ha ido pasando el documento a lo largo de su realización:

Versión	Fecha	Páginas	Procesador	Modificaciones
1	21/10/2010	18	Microsoft Word 2007	Inicio del documento, estructura del mismo creada.
1.1	10/11/2010	29	Microsoft Word 2007	Inicio del apartado de Estado del arte.
1.1 (Rev)	**18/11/2010**	**29**	**Microsoft Word 2007**	
1.2	25/11/2010	37	Microsoft Word 2007	Modificación de los puntos de la revisión de la versión 1.1.
1.2 (Rev)	**15/12/2010**	**37**	**Microsoft Word 2007**	
1.3	18/01/2011	44	Microsoft Word v	Modificación de los puntos de la revisión de la versión 1.2.
1.3 (Rev)	**20/01/2011**	**44**	**Microsoft Word 2007**	
1.4	23/02/2011	46	Microsoft Word 2007	Modificación de los puntos de la revisión de la versión 1.3.
1.4 (Rev)	**11/03/2011**	**46**	**Microsoft Word 2007**	**Finalizado del estado del arte.**
1.4.1	11/04/2011	45	Microsoft Word 2007	Modificación de los puntos de la revisión de la versión 1.4.
1.5	27/04/2011	45	Microsoft Word 2007	Inicio del apartado de Introducción, e inclusión de los requisitos en el documento
2.0	21/09/2011	102	Microsoft Word 2007	Inicio de la fase de desarrollo.
2.0 (Rev)	**19/10/2011**	**102**	**Microsoft Word 2007**	
2.0.1	27/10/2011	172	Microsoft Word 2007	Modificación de los puntos de la revisión de la versión 2.0. Se acaba la parte de la solución y la fase de desarrollo.
2.1	31/10/2011	207	Microsoft Word 2007	Apartado de evaluación finalizado y guía de usuario

				completada
2.1 (Rev)	**18/11/2011**	**207**	**Microsoft Word 2007**	
2.2	27/11/2011	221	Microsoft Word 2007	Modificación de los puntos de la revisión de la versión 2.1. Realización del apartado de gestión de proyecto, conclusiones y finalización del anexo.
2.2 (Rev)	**18/11/2011**	**221**	**Microsoft Word 2007**	
2.3	19/12/2011	224	Microsoft Word 2007	Modificación de los puntos de la revisión de la versión 2.2
2.3 (Rev)	**12/11/2011**	**224**	**Microsoft Word 2007**	
2.4	12/01/2012	224	Microsoft Word 2007	Modificación del Resumen del documento e inclusión del Abstract.

Tabla 75: Estado del documento

Anexo II. Seguimiento del proyecto fin de carrera

Para llevar a cabo las distintas tareas de la forma más eficiente posible, es necesario realizar una planificación mediante la cual organizar el tiempo del cual disponemos.

Es importante remarcar que la realización de este trabajo se hace con una limitación importante de horarios dada la situación actual del autor, por lo que la planificación inicial, que es la que expondremos en este anexo ha sufrido variaciones considerables a lo largo de la realización del proyecto.

Forma de seguimiento

Para un correcto seguimiento del proyecto, se han ido realizando entregas periódicas del documento y de la aplicación a lo largo del desarrollo de la aplicación, para obtener una posterior validación de los tutores del proyecto.

Por otro lado, para aclaración de dudas, toma de requisitos, presentación de prototipos, etc… se han ido realizando reuniones de seguimiento con los tutores.

Planificación inicial

A continuación se muestra una tabla con la planificación inicial realizada sobre las distintas fases del proyecto.

Tarea		Fecha de inicio	Fecha de fin
Estudio preliminar	Estado del arte	01/11/2010	15/12/2010
	Esbozo de la solución	01/11/2010	15/12/2010
Análisis	Definición de requisitos	01/01/2011	10/01/2011
	Especificación de requisitos	11/01/2011	31/01/2011
Diseño	Diseño modelo de datos	01/02/2011	15/02/2011
	Diseño interfaz de usuario	01/03/2011	31/03/2011
	Diseño lógica de negocio	15/02/2011	28/02/2011
Implementación	Implementación	01/04/2011	30/06/2011
Pruebas	Pruebas	01/07/2011	31/07/2011

Tabla 76: Planificación inicial del desarrollo

Por otro lado, introduciremos en la planificación también los tiempos y fechas estimados de realización de este documento.

Tarea	Fecha de inicio	Fecha de fin
Introducción	01/09/2011	10/09/2011
Estudio del problema	01/11/2010	31/12/2010
Gestión del proyecto software	01/08/2011	20/08/2011
Solución: Descripción / Fases del desarrollo	01/01/2011	30/06/2011
Evaluación	01/07/2011	31/07/2011
Conclusiones	01/09/2011	10/09/2011
Anexos	01/11/2010	10/09/2011

Tabla 77: Planificación inicial del documento

Podemos decir en este caso, que el tiempo estimado de realización del proyecto es de un total aproximado de 12 meses, iniciando el mismo en Noviembre de 2010 y finalizándolo en Noviembre de 2011.

Planificación final

A continuación se muestra una tabla con la planificación final real de las distintas fases del proyecto.

Tarea		Fecha de inicio	Fecha de fin
Estudio preliminar	Estado del arte	21/10/2010	11/03/2011
	Esbozo de la solución	21/10/2010	11/03/2011
Análisis	Definición de requisitos	29/03/2011	03/06/2011
	Especificación de requisitos	03/06/2011	27/06/2011
Diseño	Diseño modelo de datos	30/06/2011	01/09/2011
	Diseño interfaz de usuario	04/07/2011	01/10/2011
	Diseño lógica de negocio	30/06/2011	01/10/2011
Implementación	Implementación	15/06/2011	01/09/2011
Pruebas	Pruebas	15/06/2011	27/10/2011

Tabla 78: Planificación final del desarrollo

Por último presentaremos la planificación final los tiempos de realización de este documento, pues en la mayor medida posible se ha ido realizando junto con el desarrollo de la herramienta.

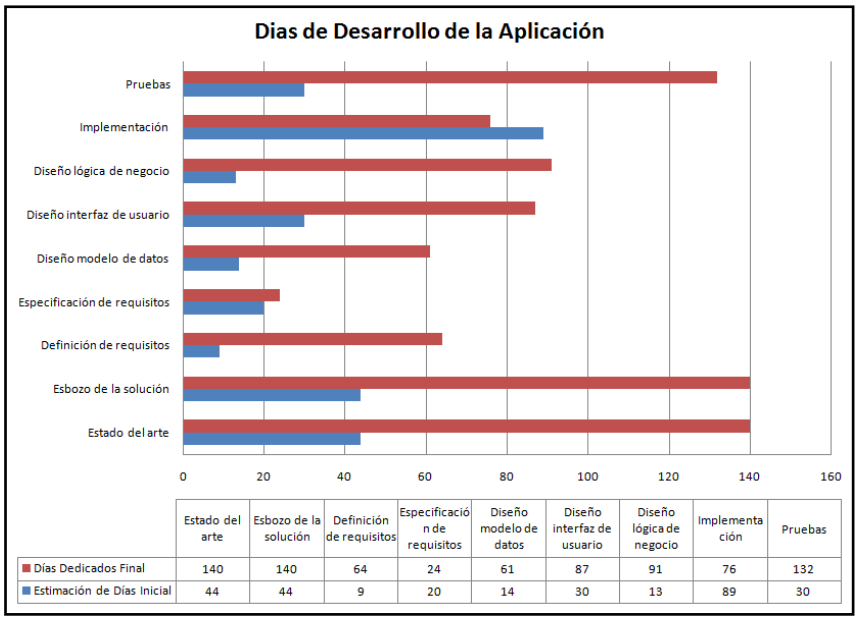
Tarea	Fecha de inicio	Fecha de fin
Introducción	27/04/2011	21/06/2011
Estudio del problema	21/10/2010	11/03/2011
Gestión del proyecto software	16/10/2011	27/11/2011
Solución: Descripción / Fases del desarrollo	21/09/2011	27/10/2011
Evaluación	20/10/2011	31/10/2011
Conclusiones	20/11/2011	27/11/2011
Anexos	21/10/2010	27/11/2011

Tabla 79: Planificación final del documento

Como se aprecia existe una desviación de todos los tiempos con respecto a la planificación inicial expuesta en el apartado anterior de este anexo. A parte de tener unos horarios muy condicionados por cuestiones laborales, la aplicación, al ser una herramienta que a posteriori va a ser usada en los laboratorios de la universidad, ha ido cambiando a lo largo de estos 12 meses, intentando cubrir en todo momento las necesidades que se iban presentando.

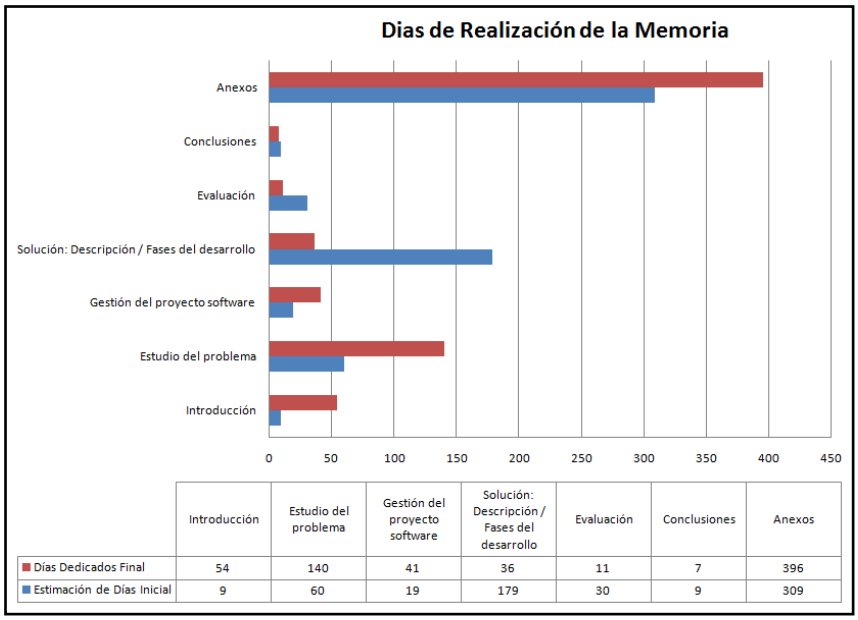
Comparativa

En las siguientes graficas se presentan las variaciones en tiempos que han sufrido cada una de las fases de desarrollo de la aplicación.



	Estado del arte	Esbozo de la solución	Definición de requisitos	Especificación de requisitos	Diseño modelo de datos	Diseño interfaz de usuario	Diseño lógica de negocio	Implementación	Pruebas
Días Dedicados Final	140	140	64	24	61	87	91	76	132
Estimación de Días Inicial	44	44	9	20	14	30	13	89	30

Figura 68: Desviación en la realización de la aplicación



	Introducción	Estudio del problema	Gestión del proyecto software	Solución: Descripción / Fases del desarrollo	Evaluación	Conclusiones	Anexos
Días Dedicados Final	54	140	41	36	11	7	396
Estimación de Días Inicial	9	60	19	179	30	9	309

Figura 69: Desviación en la realización de la memoria

Anexo III. Manual de usuario (MU)

La guía de usuario que se adjunta en este anexo del documento va dirigida a todos los potenciales usuarios de la herramienta ColDes.

Descripción Global del Sistema

La aplicación ColDes es una herramienta que da soporte a una colaboración distribuida, mediante el uso de una pizarra colaborativa.

El sistema compone de tres grandes bloques, por un lado está el modulo de usuarios, por otro lado el modulo de salas y por último el modulo de administración que permite gestionar todos los elementos definidos en el sistema.

En esta guía veremos los siguientes puntos:

- Per files del sistema.
- Modulo de administración.
- Modulo de usuarios.
- Modulo de salas.

Por último, es necesario comentar que la aplicación se ha desarrollado en dos idiomas, pudiendo cambiar el idioma del sistema en cualquier momento desde los botones situados en la parte superior derecha de la ventana.

Figura 70: MU - Botones de Idiomas

***Nota**: A lo largo de la guía, para evitar sobrecargar de imágenes innecesarias el manual, mostraremos las pantallas desde un perfil de administrador-diseñador, comentando los cambios que sufren las mismas, en caso de que existan, si el usuario es diseñador únicamente.*

Descripción de la aplicación

Usuarios del sistema

Los usuarios pueden registrarse en el sistema desde la pantalla principal de la aplicación. Además, es desde esta pantalla, desde la cual también puede acceder al sistema introduciendo su nombre de usuario y su contraseña personal, tal y como se describe posteriormente en el apartado "Identificación de usuarios".

Todos los usuarios del sistema se pueden englobar en dos perfiles distintos no excluyentes: administradores y diseñadores.

- Los ***administradores*** tienen acceso al modulo de gestión mediante el cual pueden administrar los distintos elementos presentes en el sistema: usuarios y salas.
- Los ***diseñadores*** tienen acceso al modulo de usuario, desde el cual podrán cambiar y visualizar sus datos personales, y al modulo de salas, desde el cual podrán acceder a las salas en las cuales se encuentran participando o buscar nuevas salas publicas creadas en el sistema.

Por otro lado, todos los usuarios que participan en las salas tienen una determinada función en cada una de ellas. Estas funciones en las salas pueden ser: propietario, moderador, colaborador e invitado. En las secciones posteriores explicaremos los permisos que tendrán estos usuarios dentro de la sala.

Identificación de usuarios

Como se ha especificado en el apartado anterior, para acceder al sistema, el usuario ha de introducir en el componente de autenticación su nombre de usuario y la contraseña personal del mismo. Es necesario que los datos que se introduzcan se correspondan con los datos almacenados en base de datos, y que el usuario que intenta acceder este activado dentro del sistema. La activación de los usuarios dentro del sistema la tiene que realizar un usuario cuyo perfil sea de administrador.

Los usuarios podrán acceder a la aplicación mediante la siguiente url:

http://<server_name>:8400/ColDes/ColDes.html

Figura 71: MU - Pantalla de Acceso

En caso de que el usuario introduzca el nombre de usuario o la contraseña de forma incorrecta, el sistema avisara al usuario para que vuelva a introducir los datos mostrándole el siguiente error:

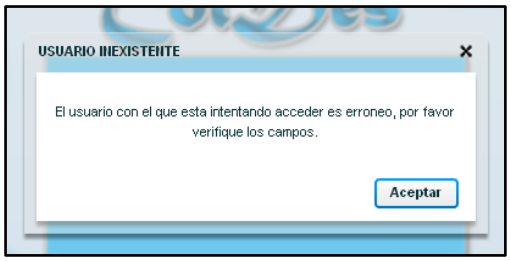


Figura 72: MU - Error de autenticación

Este mismo mensaje le saldrá a los usuarios que aún no hayan sido activados en el sistema por un administrador del mismo desde el módulo de gestión de usuarios que veremos más adelante.

En caso de que el usuario logre autenticarse correctamente en el sistema, automáticamente obtendrá una sesión activa que tendrá una duración no superior a 30 minutos, en los cuales, si el usuario ha estado inactivo, la sesión pasará a no ser válida. Tras esto el usuario ya se encontrara en la pantalla de inicio del sistema, desde la cual podrá acceder a los distintos menús definidos en la aplicación.

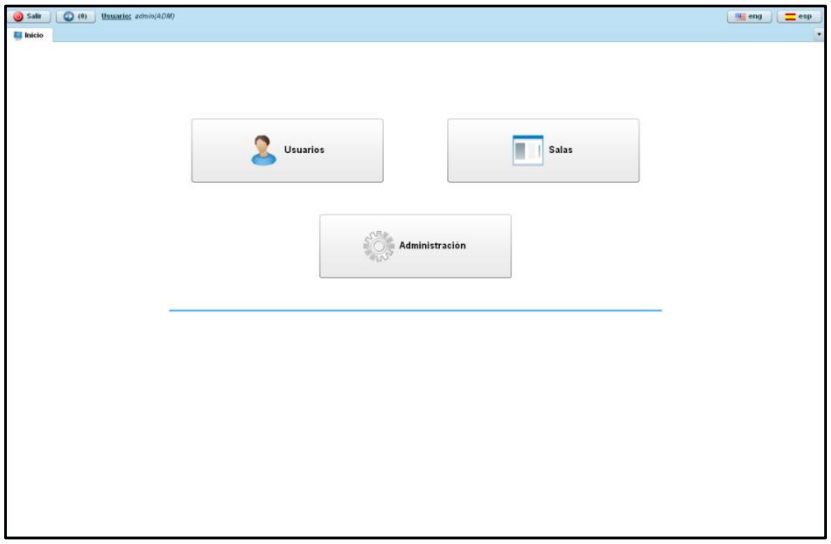


Figura 73: MU - Pantalla de Inicio

Esta pantalla de inicio varía según el perfil que tenga el usuario que ha accedido al sistema; por un lado el usuario administrador tendrá visibilidad del menú de Administración, mientras que el usuario que únicamente es diseñador no podrá verlo.

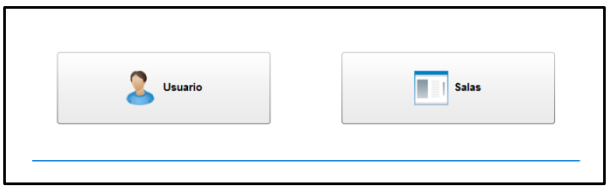


Figura 74: MU - Menús inicio del usuario diseñador

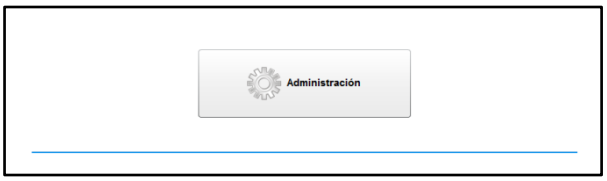


Figura 75: MU - Menús inicio del usuario administrador

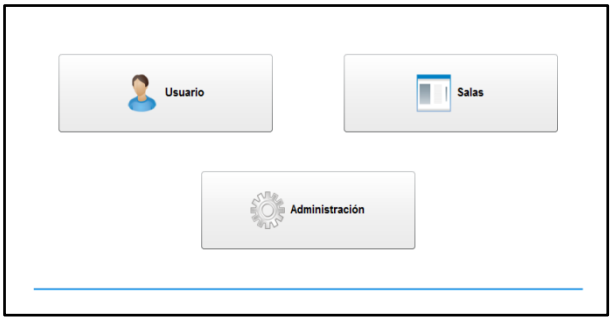


Figura 76: MU - Menús inicio del usuario administrador - diseñador

Una vez que el usuario termine de usar la aplicación, deberá desconectarse del sistema usando el botón de cerrado de sesión presente en la parte superior derecha de la pantalla, que es visible en todo momento una vez que el usuario se ha autenticado en el sistema. Si bien este paso no es obligatorio si es recomendable.

Figura 77: MU - Botón para salir de la aplicación

Tras esto, el usuario se encontrara en la pantalla de autenticación del sistema.

Registro de usuarios

Cualquier persona que acceda a la aplicación mediante la url comentada en el apartado anterior puede registrarse en el sistema desde la pantalla de autenticación del sistema.

Figura 78: MU - Registro de usuario

Para ello el usuario deberá rellanar el formulario correspondiente aportando todos los datos que este solicita y enviar la solicitud de registro.

Figura 79: MU - Formulario de registro de usuarios

Una vez enviado el formulario, el usuario ha de esperar a que algún administrador del sistema le active el usuario desde el modulo de gestión de usuario de la aplicación. En caso de intentar acceder con un usuario inactivo, el sistema considerara al usuario como no existente, y denegara el acceso a la aplicación.

Modulo de administración

Desde este modulo, el usuario podrá gestionar tanto los usuarios como las salas definidas en el sistema. Este modulo únicamente será visible para aquellos usuarios que tengan el perfil de administrador y podrán acceder a él desde el menú principal de la aplicación.

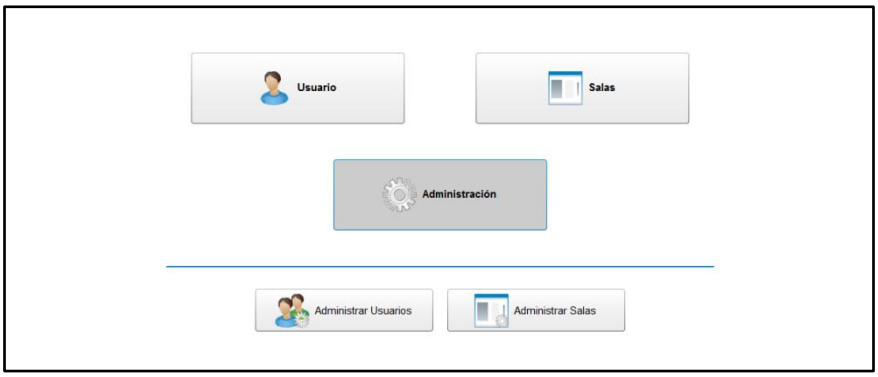


Figura 80: MU - Submenús Administración

Como hemos comentado anteriormente, desde este menú podemos acceder a la gestión de los distintos elementos del sistema: usuarios y salas. A continuación explicaremos las distintas opciones que tienen los usuarios administradores desde estos menús de administración.

Administración de usuarios

Desde esta sección de la aplicación los usuarios administradores pueden gestionar los distintos usuarios que están registrados en la aplicación.

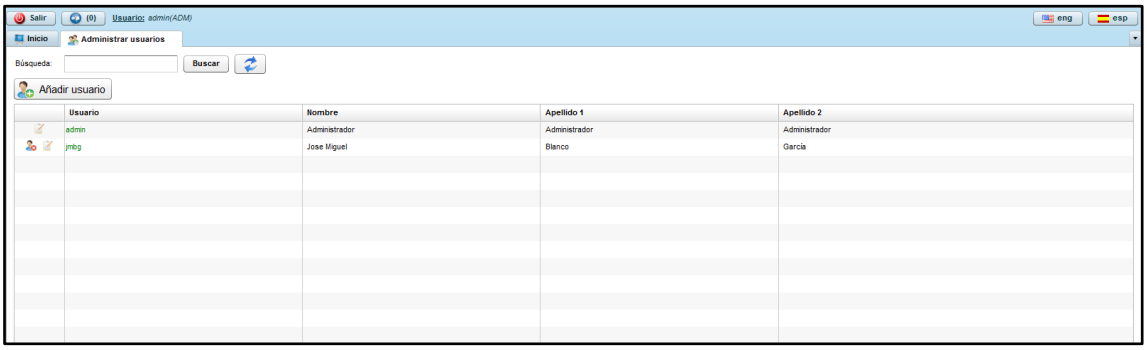


Figura 81: MU - Adminitración de usuarios

El color que se presenta con el nombre de usuario del usuario nos indica los usuarios activos e inactivos del sistema. El color verde indica que el usuario esta activo. Por el contrario, cuando un usuario esta desactivado, el color del nombre de usuario del usuario aparecerá en rojo.

El administrador podrá realizar búsquedas sobre los distintos usuarios del sistema introduciendo un patrón en el cuadro de búsqueda de la parte superior de la tabla. El texto introducido se compara con el nombre de usuario, nombre real y apellidos, y en caso de coincidencia, la tabla mostrara únicamente aquellos registros que cumplan la cadena de búsqueda.

Figura 82: MU - Formulario de búsqueda

Desde esta ventana el administrador podrá gestionar los usuarios mediante los submenús que aparecen en el lado izquierdo de cada usuario  :

-  Mediante este botón el administrador podrá elimina al usuario del sistema. A la hora de borrar un usuario el sistema preguntará al administrador si realmente desea borrar al usuario.

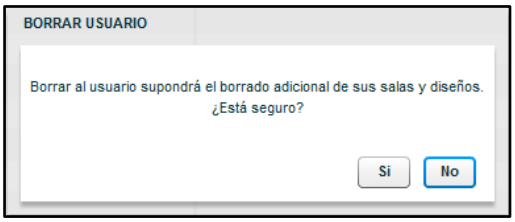


Figura 83: MU - Confirmación de borrado de usuario

-  Mediante este botón el administrador podrá acceder a los datos personales del usuario. De esta forma, se abrirá una ventana emergente en la cual aparecerán los datos del usuario.

Figura 84: MU - Formulario de datos de usuario

Desde esta pantalla el administrador puede modificar todos los datos del usuario con excepción del nombre de usuario. Puede realizar modificaciones sobre los datos personales del usuario, asignarle cualquiera de los perfiles definidos en el sistema y activarlos o desactivarlos.

Por último, los usuarios administradores podrán crear nuevos usuarios desde esta ventana a través del botón situado en la parte superior de la ventana "*Añadir Usuario*".

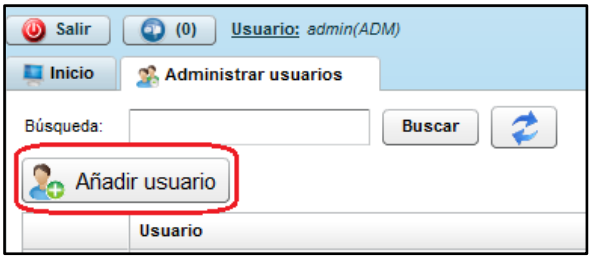


Figura 85: MU - Añadir un nuevo usuario

Una vez pulsado este botón se abrirá una pantalla de registro de usuario análoga a la que se presenta al usuario normal en la pantalla de autenticación a la hora de registrar un nuevo usuario.

Administración de salas

Desde esta sección de la aplicación los usuarios administradores pueden gestionar las distintas salas creadas por los usuarios del sistema.

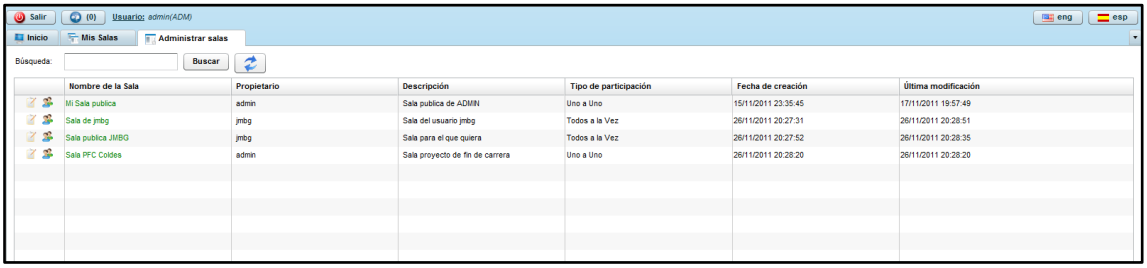


Figura 86: MU - Administración de salas

De igual forma que en la gestión de usuarios, y en el resto de ventanas de búsqueda, el usuario podrá realizar búsqueda sobre las distintas salas mostradas en la tabla. En este caso el patrón de búsqueda se comparara con el nombre de la sala y con el nombre de propietario de la sala.

El color del nombre de la sala indica si esta se encuentra abierta o cerrada. Verde nos indica que la sala se encuentra actualmente abierta, mientras que rojo nos indicaría que la sala en este momento está cerrada, impidiendo de esta forma la participación en la misma.

Mediante los iconos situados a la izquierda de cada una de las salas, el administrador puede gestionar tanto los datos de las propias salas como lo usuarios que participan o pueden participar en las mismas  .

- Mediante el icono  el usuario podrá gestionar los datos de la propia sala, pudiendo modificar cualquiera de ellos: nombre, visibilidad de la sala en el sistema (pública o privada), estado de la sala, tipo de participación y descripción. Al hacer click en este icono para obtener los datos de la sala, se abrirá una ventana emergente mediante la cual el usuario podrá ver y modificar los datos de la sala.

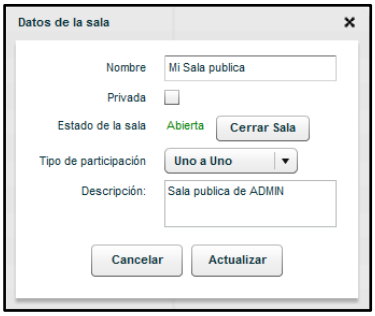


Figura 87: MU - Formulario de datos de salas

- Mediante el icono  el usuario podrá gestionar por un lado los usuarios que están participando, pudiendo modificar la función que realizan cada uno de ellos dentro de la sala o expulsarlos de la misma, y por otro lado enviar invitaciones a los usuarios del sistema que no están participando ya en la sala.

Figura 88: MU - Administración de usuarios de sala

A continuación se muestran las distintas acciones que el usuario administrador puede realizar desde esta ventana:

- Gestión de usuarios de la sala:

- Se podrá cambiar la función de cada uno de los usuarios de la sala mediante el icono  . Al acceder a los datos mediante este icono se desplegaran en la zona inferior de la ventana los datos del usuario dentro de la sala.

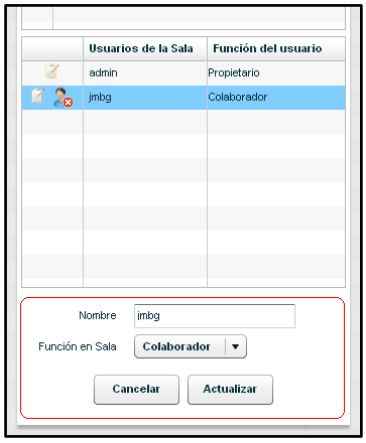


Figura 89: MU - Modificación de la función de un usuario

- El icono  permitirá expulsar de la sala al usuario.

- Invitación de usuarios nuevos a la sala mediante el icono  . A la hora de invitar a un nuevo usuario a la sala, una ventana emergente le pedirá al usuario que seleccione una función con la cual desea que el usuario invitado participe dentro de la sala.

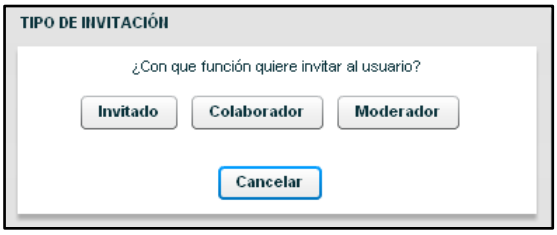


Figura 90: MU - Selección de función en la invitación

Modulo de Usuario

Desde este modulo, cualquier usuario podrá acceder a sus datos personales para consultarlos o modificarlos y a los diseños guardados por el usuario.

Figura 91: MU - Submenús Usuario

Mis Datos

Cuando el usuario pulsa el botón "*Mis Datos*" se abrirá una ventana emerger gente donde los usuarios puede visualizar sus datos personales y modificarlos si corresponde, a excepción del nombre de usuario que no se podrá modificar en ningún momento desde la aplicación.

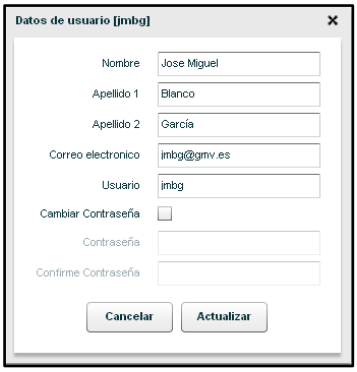


Figura 92: MU - Datos de usuario

Por otro lado, en caso de que el usuario sea administrador también tendrá acceso a los datos de perfiles y de activación de su propio usuario.

Figura 93: MU - Datos extra del usuario administrador

Mis Diseños

Cuando el usuario pulsa el botón "*Mis Diseños*" se abrirá una nueva pestaña donde el usuario pude visualizar cada uno de los diseños que este tenga almacenados.

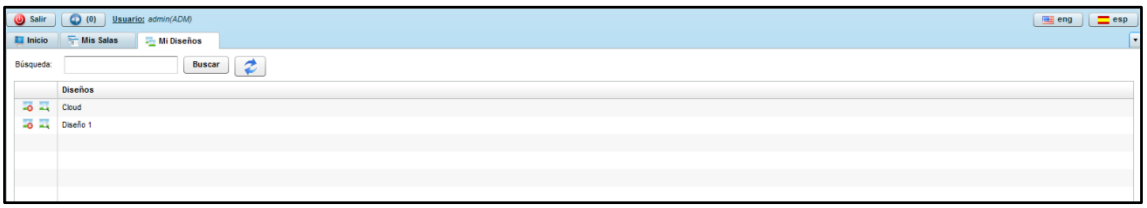


Figura 94: MU - Mis diseños

Desde esta pantalla el usuario podrá previsualizar los diseños que tiene guardados y eliminarlos mediante los botones situados en el lado izquierdo de cada uno de los diseños del usuario  .

-  Mediante este botón el usuario eliminara el diseño de sus diseños. El sistema preguntara al usuario, a modo de confirmación, si realmente quiere borrar el diseño.

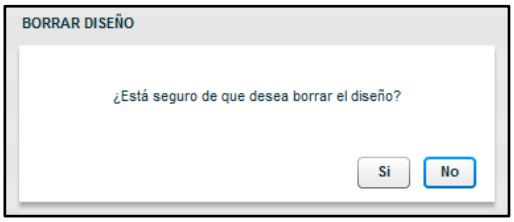


Figura 95: MU - Confirmación de borrado de diseño

-  Mediante este botón el usuario podrá previsualizar el diseño en la parte inferior de la pantalla.

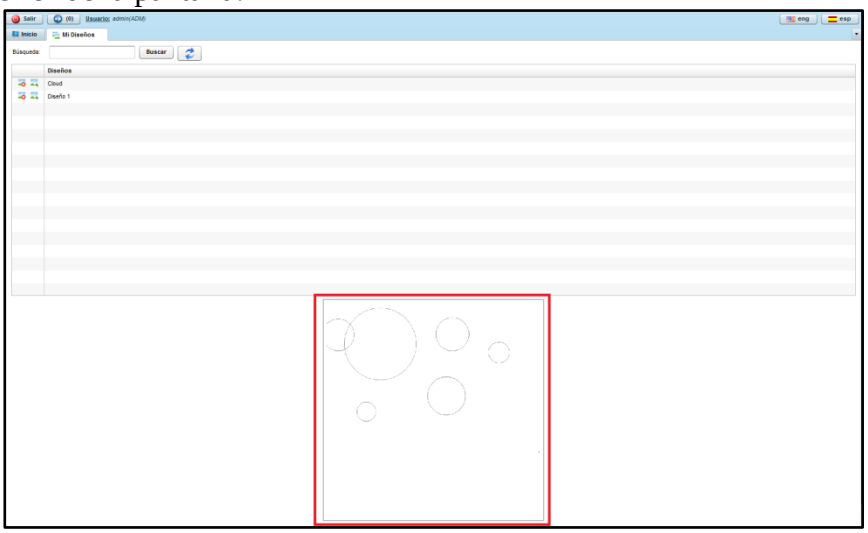


Figura 96: MU - Previsualización de diseños

Modulo de Salas

Desde este modulo, cualquier usuario podrá acceder a las distintas salas creadas en el sistema. Un usuario puede consultar las salas en las cuales se encuentra participando, o buscar nuevas salas en las que participar.

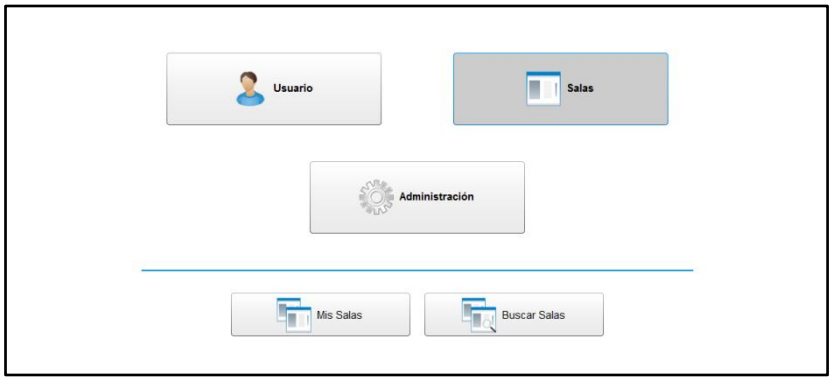


Figura 97: MU - Submenús Salas

A continuación explicaremos las distintas opciones que tienen los usuarios desde estos menús de salas.

Buscar Salas

Desde esta sección de la aplicación los usuarios pueden buscar nuevas salas en las que participar con la función de colaboradores. Las salas que podrán encontrar desde este buscador únicamente son salas definidas como publicas en el sistema y aquellas en las cuales el usuario no se encuentre participando ya, para participar en las salas privadas del sistema es necesario que un administrador, propietario de sala o moderador de sala, envíe una invitación al usuario.

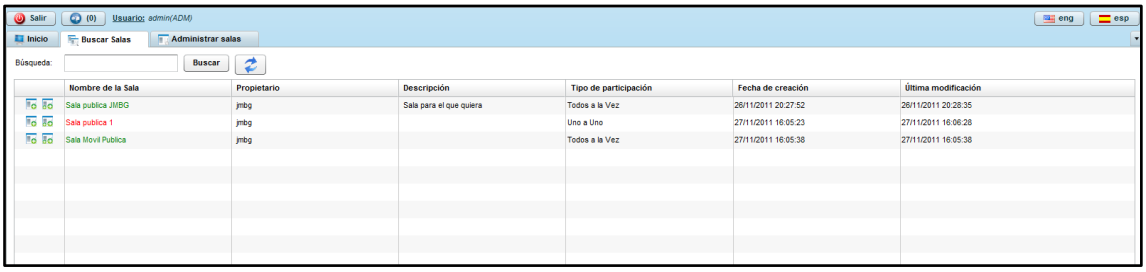


Figura 98: MU - Búsqueda de salas

El color del nombre de la sala indica si esta se encuentra abierta o cerrada. Verde nos indica que la sala se encuentra actualmente abierta, mientras que rojo nos indicaría que la sala en este momento está cerrada, impidiendo de esta forma la participación en la misma.

Las búsquedas que el usuario puede realizar en esta pantalla del sistema es equivalente a la búsqueda que se puede hacer en la pantalla de gestión de salas. El

patrón introducido en el buscador se comparara con el nombre de la sala, el nombre de usuario del propietario y la descripción de la sala.

Desde esta ventana el usuario podrá acceder directamente a la sala, o únicamente apuntarse a ella, para ello se habilitan los botones situados en la parte izquierda de los datos de la sala  :

-  Mediante este botón el usuario podrá apuntarse únicamente a la sala. Con esta acción la sala desaparecerá de la pantalla de búsquedas y a partir de ese momento podrá acceder a la misma desde el menú de "*Mis Salas*".
-  Mediante este botón el usuario podrá apuntarse y acceder directamente a la sala. Con esta acción la sala desaparecerá de la pantalla de búsquedas y a partir de ese momento podrá acceder a la misma desde el menú de "*Mis Salas*".

Independientemente de la opción que use el usuario para comenzar a participar en una sala, su función dentro de la misma será de colaborador.

Mis Salas

Desde esta sección de la aplicación los usuarios pueden ver las salas en las cuales se encuentran participando, gestionar aquellas salas sobre las cuales tengan privilegios suficientes, dejar de participar en las mismas y crear nuevas salas.

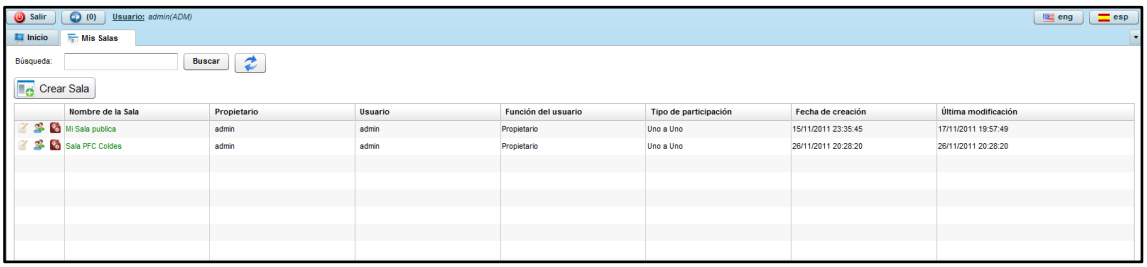


Figura 99: MU - Mis salas

Desde aquí el usuario puede realizar distintas acciones sobre las salas en las cuales está participando, en relación a la función que el usuario desempeña dentro de la sala. Las acciones que se pueden realizar sobre cada sala son las siguientes:

- Mediante el icono  el usuario podrá gestionar los datos de la propia sala, pudiendo modificar cualquiera de ellos: nombre, visibilidad de la sala en el sistema (pública o privada), tipo de participación y descripción. Al hacer click en este icono para obtener los datos de la sala, se abrirá una ventana emergente mediante la cual el usuario podrá ver y modificar los datos de la sala.

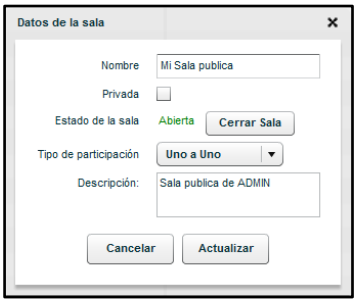


Figura 100: MU - Formulario de datos de salas

- Mediante el icono  el usuario podrá gestionar por un lado los usuarios que están participando, pudiendo modificar la función que realizan cada uno de ellos dentro de la sala o expulsarlos de la misma, y por otro lado enviar invitaciones a los usuarios del sistema que no están participando ya en la sala.

Figura 101: MU - Administración de usuarios de sala

  - Las acciones que el usuario puede realizar desde esta ventana son las mismas a las que puede realizar un usuario administrador desde la parte de gestión de salas del sistema.

- Mediante el icono  el usuario puede dejar de participar en la sala. Esta acción borra la relación existente entre el usuario y la sala, pero en caso de que el usuario sea el propietario de la sala, el sistema avisara a este para notificarle que dejar de participar en la sala de la cual es el propietario implicara el borrado de la sala y de todas las relaciones de los usuarios con la misma.

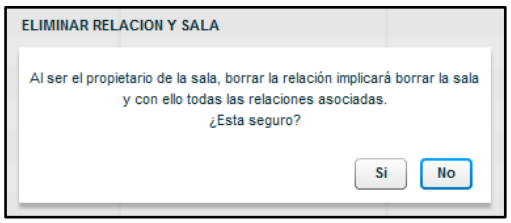


Figura 102: MU - Confirmación borrado de sala

Como hemos comentado, según la función que el usuario tenga dentro de la sala, este podrá realizar unas acciones u otras en lo referente a la gestión de la sala, a continuación mostramos según las distintas funciones del sistema dentro de las salas, que permisos tienen sobre las salas:

- Propietario:  .
- Moderador:  .
- Colaborador / Invitado:  .

Por otro lado, desde esta ventana, el usuario puede crear salas mediante el botón definido en la parte superior de la ventana debajo del formulario de búsqueda "*Crear Sala*". El sistema mostrara al usuario el siguiente formulario de creación de nueva sala, donde solicitara al usuario: nombre de la sala, visibilidad de la sala, tipo de participación y una breve descripción.

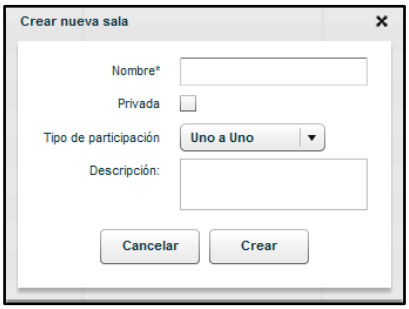


Figura 103: MU - Registro de nueva sala

Cuando un usuario crea una nueva sala, su función dentro de la misma será la de propietario

Dentro de una sala

El acceso a las salas de ColDes se puede realizar desde la ventana de búsqueda de salas o desde la ventana de mis salas. En cualquier caso al acceder a una sala el usuario podrá ver el lienzo, el chat, las distintas acciones que este puede realizar dentro de la sala según la función que desempeñe dentro de la misma y el log de acciones de la sala.

Dependiendo de la forma de participación de la sala, esta tendrá los botones de solicitud de pincel o no.

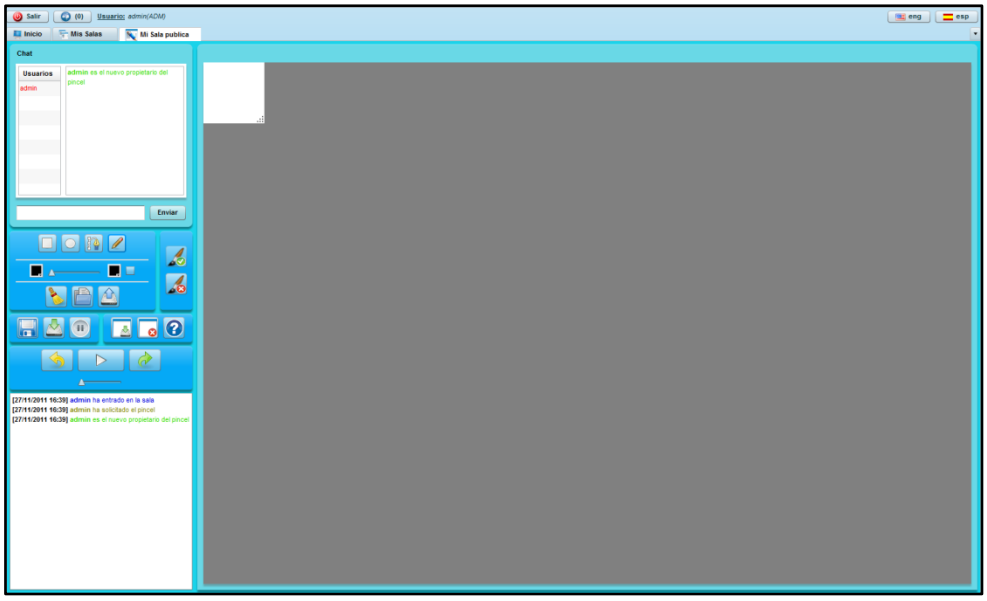


Figura 104: MU - Ventana de Sala (Uno a Uno)

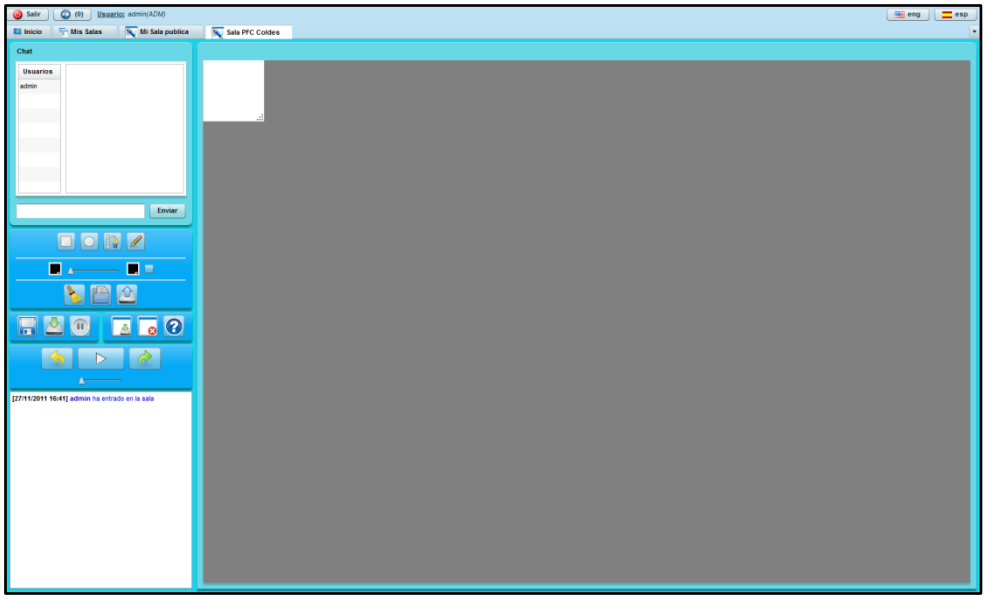


Figura 105: MU - Ventana de Sala (Todos a la vez)

Como hemos comentado, dependiendo de la función que el usuario tenga dentro de la sala, podrá visualizar un conjunto de botones u otro.

Figura 106: MU - Acciones según función del usuario

A continuación explicaremos la funcionalidad de cada conjunto de botones:

Estos botones permiten al usuario interactuar con el lienzo de la sala, bien sea dibujando elementos sobre el lienzo, configurar parámetros de color, grosor, relleno de las figuras, limpiar el lienzo, y guardar el contenido del lienzo.
Estas acciones únicamente las pueden usar los usuarios cuya función sea: propietario, moderador o colaborador.

ELEMENTO	FUNCION
	Permite al usuario pintar cuadrados dentro del lienzo.
	Permite al usuario pintar círculos dentro del lienzo.
	Permite al usuario pintar rectas dentro del lienzo.
	Permite al usuario dibujar a mano alzada dentro del lienzo.
	Permite al usuario seleccionar el color y el grosor del borde de la figura que este dibujando.
	Permite al usuario seleccionar si desea relleno en la figura que está dibujando, y el color de este.
	Permite al usuario limpiar el lienzo de la sala. Elimina todos los elementos pintados sobre el mismo.
	Permite al usuario importar una imagen de PC al lienzo de la sala.
	Permite al usuario importar un diseño del usuario al lienzo de la sala.

Estos botones permiten al usuario interactuar con los elementos que permiten reproducir las distintas acciones que se han ido realizando sobre el lienzo desde que el usuario entro en la sala.
Estas acciones pueden ser usadas por cualquier usuario independientemente de la función que desempeñen en la sala.

ELEMENTO	FUNCION
	Permite al usuario deshacer las acciones realizadas sobre el lienzo.
	Permite al usuario rehacer acciones ya realizadas sobre el lienzo.
	Permite al usuario ver en forma de secuencia las distintas acciones que se han ido realizando sobre el lienzo, desde el principio al punto actual.
	Especifica la diferencia en segundos entre secuencia y secuencia a la hora de ejecutar el play descrito antes.

Estos botones permiten al usuario guardar el contenido del lienzo, bien como un diseño del usuario o bien como una imagen exportada a fichero local. de color, grosor, relleno de las figuras, limpiar el lienzo, y guardar el contenido del lienzo.
Estas acciones pueden ser usadas por cualquier usuario independientemente de la función que desempeñen en la sala.

ELEMENTO	FUNCION
	Permite al usuario exportar el contenido del lienzo a una imagen en el PC del usuario.
	Permite al usuario exportar el contenido del lienzo a base de datos como un nuevo diseño del usuario.
	Permite al usuario para la sincronización de datos a la hora de recibir las actualizaciones de las acciones que realizan el resto de usuarios dentro del lienzo.

Estos botones permiten al usuario interactuar con el pincel de la sala, hacer peticiones de uso de pincel y soltarlo en el caso de que lo tengan asignado. Estos botones solo son visibles cuando el tipo de participación en la sala es por turnos (uno a uno).
Estas acciones únicamente las pueden usar los usuarios cuya función sea: propietario, moderador o colaborador.

ELEMENTO	FUNCION
	Permite al usuario solicitar el pincel para poder pintar en el lienzo.
	Permite al usuario dejar el pincel para que la siguiente persona que solicito el pincel pueda pintar.

Estos botones permiten al usuario interactuar con la persistencia de diseños dentro de la sala y evaluar la opinión de los usuarios participantes con respecto al contenido del lienzo.
Estas acciones únicamente las pueden usar los usuarios cuya función sea: propietario o moderador.

ELEMENTO	FUNCION
	Permite al usuario asociar el contenido del lienzo actual a la sala, para que de esta forma al entrar de nuevo en la misma el lienzo por defecto sea el diseño asociado.
	Permite al usuario eliminar el diseño asociado a la sala.
	Permite al usuario lanzar una encuesta a los usuarios que se encuentra colaborando en el interior de la sala. Las encuestas pueden ser para asociar el diseño a la sala o para eliminar el diseño de la sala.

Colaboración en las salas

Como hemos comentado anteriormente, toda sala creada en el sistema exige que se especifique un tipo de participación dentro de la misma. Este tipo esta englobado en dos formas de participación:

- Por una lado todos a la vez, en la cual cualquier usuario que este dentro de la sala y tenga permisos para pintar dentro del lienzo puede pintar,
- Por otro lado de uno en uno, donde en este caso las acciones que se realizan dentro del lienzo están moderadas por el pincel de la sala, "*Unicamente puede pintar aquel usuario que tenga el pincel de la sala en posesión*"

La colaboración que se realiza en un sala cuyo tipo de participación es de todos a la vez, no está moderada de ningún tipo, y puede darse casos de problemas de sincronismo entre los lienzos de los distintos usuarios, es por ello que al crear una sala cuyo tipo de participación es de todos a la vez, el sistema informa al usuario de los problemas que esto puede conllevar.

El modo que vamos a explicar en este apartado del manual de usuario es el de "*Uno a uno*", donde entra el concepto de pincel como elemento moderador de la colaboración en el lienzo de la sala.

El concepto de la participación en este tipo de salas es simple, únicamente el usuario que tenga el pincel puede pintar en el lienzo. Para evitar que un usuario que no tiene el pincel pueda pintar en el lienzo, el panel que permite interactuar sobre el lienzo esta deshabilitado hasta que el usuario reciba el pincel.

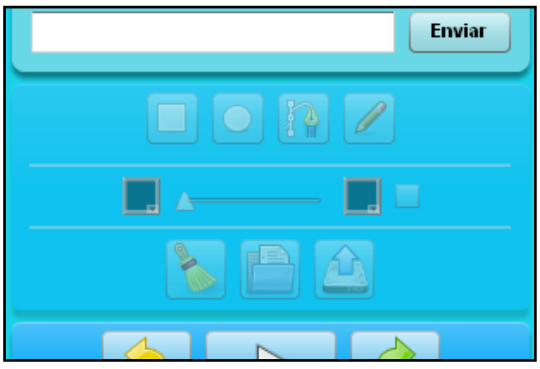


Figura 107: MU – Panel de interacción bloqueado

Todos los usuarios que tengan permiso para pintar dentro de la sala podrán solicitar el pincel en cualquier momento, lo que les pondrá en cola para usar el pincel.

Figura 108: MU - Acciones sobre pinceles

Esta cola tiene una política mediante la cual se selecciona el siguiente usuario propietario del pincel, en la versión actual de la aplicación esta política de prioridad sigue un modelo FIFO, "*First Input, First Output*", es decir, la primera petición en llegar será el siguiente propietario del pincel cuando el actual propietario deje el pincel.

Cuando un usuario solicita el pincel, en el log de la sala aparecerá un mensaje mediante el cual se notifica al resto de usuarios que un usuario ha solicitado el pincel.

Figura 109: MU - Notificación de petición de pincel

En el momento que el usuario que ha solicitado el pincel lo recibe, esto se notifica al resto de usuarios, y la paleta de acciones sobre el lienzo del nuevo propietario del pincel se activa para permitir a este dibujar en el lienzo.

Figura 110: MU - Notificación de recepción de pincel

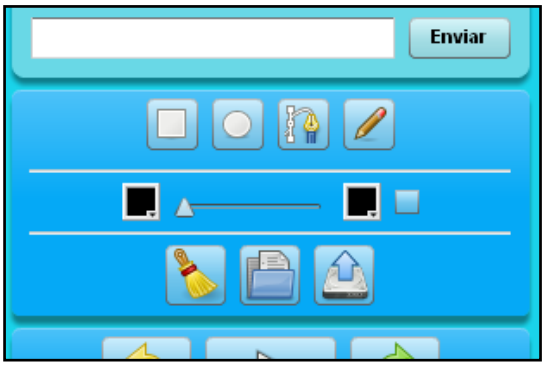


Figura 111: MU – Panel de interacción desbloqueado

Por otro lado, la paleta de acciones sobre el lienzo del usuario que ha perdido el pincel se bloquea a la espera de recibir de nuevo el pincel.

Invitaciones a las salas

Las invitaciones a participar dentro de las salas, se pueden enviar desde la ventana de gestión de salas por parte de los usuarios administradores y desde la ventana de mis salas el usuario solo puede mandar invitaciones a las salas en las cuales su función es de propietario o de moderador.

El usuario que envía la invitación ha de seleccionar la función que quiere que el usuario invitado desempeñe dentro de la sala. Tras esto, el usuario invitado recibirá una notificación en su buzón, indicándole que tiene nuevos mensajes sin leer. Este buzón se puede ver en la parte superior izquierda de la pantalla.

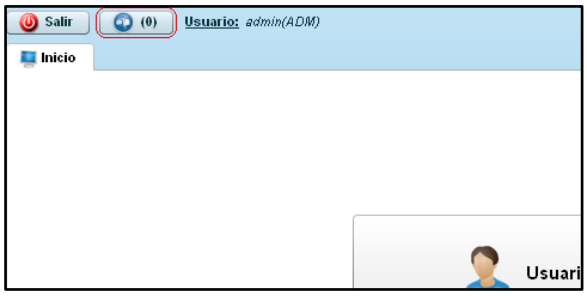


Figura 112: MU - Buzón de Invitaciones

Si el usuario está conectado cuando recibe una invitación, alado de su buzón de invitaciones aparecerá una notificación de que ha recibido una invitación nueva:

Figura 113: MU - Notificación de nueva invitación

Cuando el usuario acceda a sus mensajes, podrá ver todas las invitaciones que el usuario ha recibido. Cada invitación informara al usuario a que sala se le ha invitado, quien es el propietario de la sala y qué función va a desempeñar en la misma.

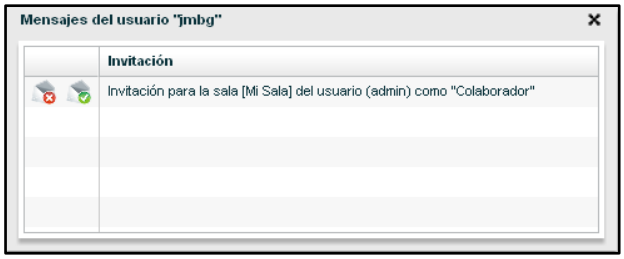


Figura 114: MU - Invitaciones de usuarios

El usuario desde esta ventana podrá aceptar ( ) o rechazar ( ) las invitaciones. En caso de rechazar una invitación de participación, esta se elimina únicamente, y en caso de aceptar la invitación, esta se elimina y se crea una relación entre el usuario y la

sala a la cual ha sido invitado. A partir de ese momento, el usuario puede ver la sala, gestionarla (si procede) y acceder a la misma, desde la ventana de "*Mis Salas*".

De forma adicional podrá observar la información de la descripción de la sala asociada a la invitación seleccionando la invitación que desea consultar.

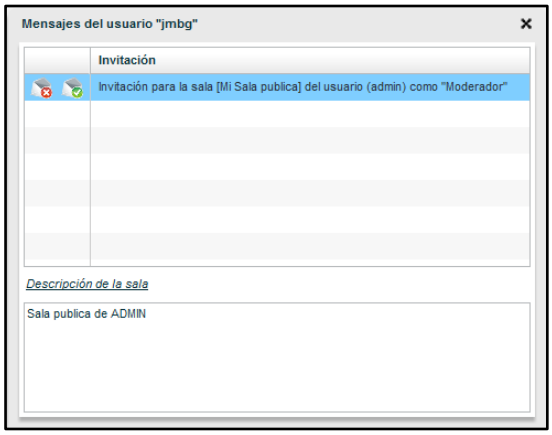


Figura 115: Descripción de la sala en la invitación

Anexo IV. Instalación del entorno

Dentro de este anexo realizaremos un pequeño tutorial de cómo se ha instalado el entorno de desarrollo en el cual se han realizado las pruebas de la aplicación.

Los pasos descritos a continuación son comunes independientemente de la plataforma en la que se desee instalar el entorno, sin embargo la explicación que se muestra a continuación es para la preparación del entorno en una plataforma Unix como Ubuntu v.10.

1. *Instalación y configuración de Java JDK.*
    a. Verificar la versión de java instalada en el servidor. Para ello, es necesario abrir un terminal y escribir ***javac***.
        i. Si el JDK está instalado el terminal mostrara la forma de uso del comando ***javac***.
        ii. En caso de no estar instalado, la terminal te indicara en que paquetes se puede encontrar el comando:

Figura 116: Paquetes con el javac

    b. En caso de tener la versión 1.5 o superior del JDK podemos saltarnos el resto de pasos de configuración y pasar directamente a la configuración de **BlazeDS**.
        i. Para saber la versión del JDK instalado simplemente hay que introducir el siguiente comando: ***java -version***
    c. Para obtener el JDK en un sistema Unix sería suficiente usar el comando ***apt-getinstall***indicando el paquete que se desea instalar, en este caso el jdk de java.
        i. ***sudo apt-get install sun-java6-jdk***

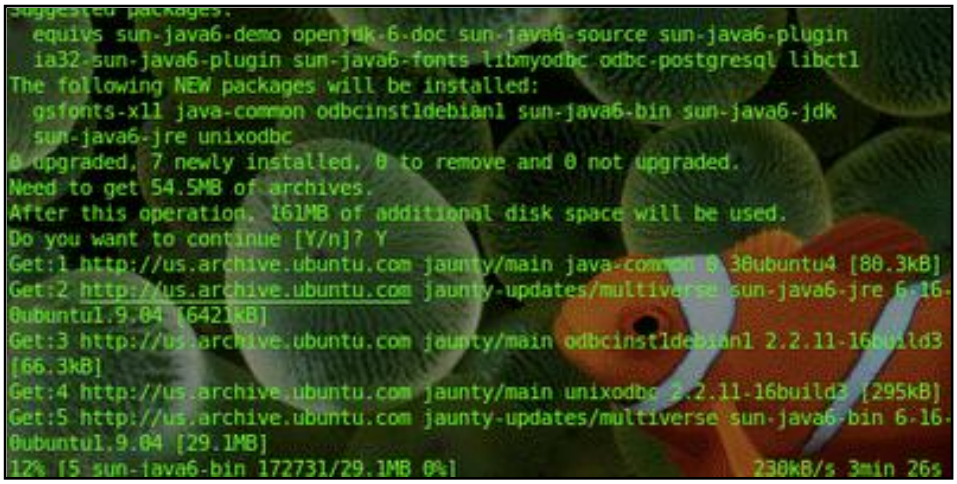


Figura 117: apt-get install java

d. Una vez que el paquete se ha instalado correctamente, podemos verificarlo escribiendo en la terminal ***javac*** y este mostrara la forma de uso del comando ***javac***.
e. Tras instalar el JDK es necesario configurar unas variables de entorno que indiquen en qué lugar del sistema de ficheros se ha instalado el JDK. En sistemas Ubuntu este directorio suele encontrarse bajo la ruta */user/lib/jvm/java-6-sun*
    i. ***export JAVA_HOME=/usr/lib/jvm/java-6-sun***
    ii. Si se desea que esta parte se realice de forma automática con el inicio de sesión del usuario es necesario introducir esta sentencia al final del fichero de configuración del usuario **~/.bashrc.**
    iii. Para verificar que se ha configurado correctamente la variable de entorno basta con escribir en una terminal el siguiente comando:***echo $JAVA_HOME***. La respuesta será la ruta que fue asignada en el punto i (***/usr/lib/jvm/java-6-sun***).

2. *Instalación y configuración de BlazeDS*
    a. La instalación de **BlazeDS** se puede resumir en los siguientes puntos:
        i. Descargar la versión*turnkey*de **BlazeDS** de la página de Adobe: http://opensource.adobe.com/wiki/display/blazeds/BlazeDS
        ii. Descomprimir el archivo y copiar el contenido en la ruta donde se desea instalar **BlazeDS**.
    b. Tras esto es necesario dar permisos de ejecución a los distintos elementos contenidos en la carpeta donde se descomprimió **BlazeDS**.
        i. ***sudo chmod -R 755 BlazeDS/*** and press Enter, then type your root password.
    c. Para iniciar la distribución de tomcat especifica de **BlazeDS**:
        i. Moverse al directorio *BlazeDS/tomcat/bin*
        ii. Ejecutar ***./catalina.sh run***
    d. Mediante un navegador podemos acceder a la página principal del *tomcat* en la siguiente url: http://localhost:8400

i. El puerto se establece por defecto en la configuración del *tomcat* de **BlazeDS:**

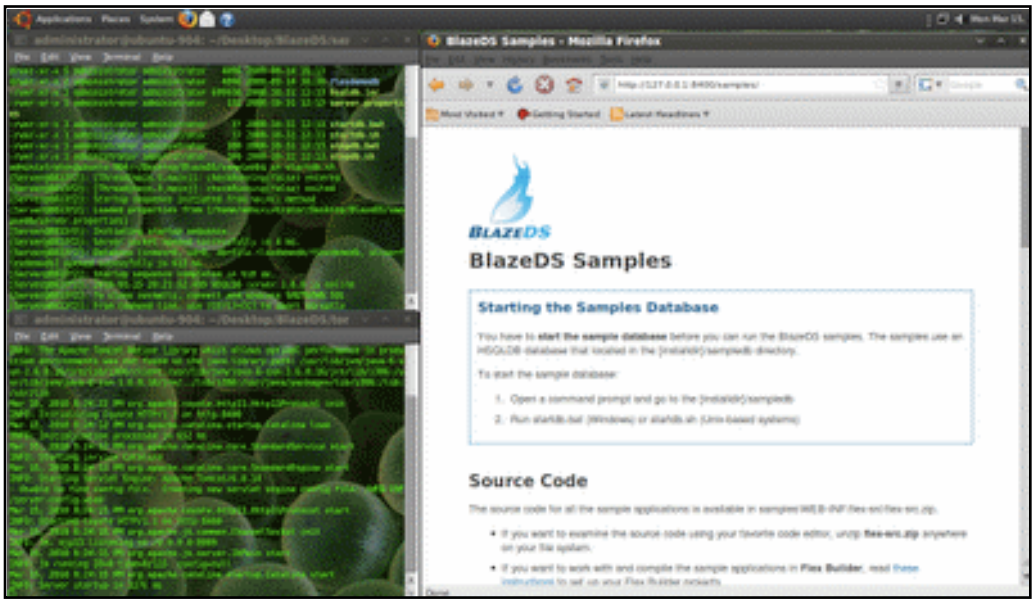


Figura 118: Pantalla de inicio de BlazeDS

3. *Instalación MySQL Server 5.1*
   a. Esta base de datos está disponible por defecto en el repositorio de programas de Ubuntu 10 y se puede instalar sencillamente mediante el siguiente comando:
      i. ***sudo apt-get install mysql-server-5.1***
   b. Después de instalar el servidor de base de datos es necesario configurarla y crear el *schema* que contendrá la base de datos del sistema.
      i. Activar MySQL con el siguiente comando:
      sudo mysql_install_db.
      ii. Acceder a la base de datos con el usuario y la contraseña del usuario *root* (estos dato se han de facilitar en la instante de inserción):
      mysql -u root –p
      iii. Una vez dentro del servidor tenemos que crear la base de datos, y un usuario para acceder a esta:
         1. Creamos la base de datos:
         CREATE DATABASE newDB;
         2. Creamos el usuario y la contraseña del usuario:
         CREATE USER newDBUs;
         SET PASSWORD FOR newDBUs = PASSWORD(“password”);
         3. Asignamos permisos al usuario sobre la base de datos:
         GRANT ALL PRIVILEGES ON newDB.*
         TO newDBUs IDENTIFIED BY ‘password’;
   c. Si quisiéramos restaurar la base de datos a partir de un backup de la misma podríamos hacerlo mediante el siguiente comando.
      i. ***mysql -u root -p nombre_base_de_datos < fichero.sql***

4. *Despliegue de aplicaciones*
    a. Para desplegar una aplicación en el *tomcat* hay que seguir los siguientes pasos:
        i. Detener el servidor: ***./catalina.sh stop***
        ii. Copiar en la carpeta *BlazeDS/tomcat/webapp* el ***.war*** generado de la aplicación
        iii. Arrancar de nuevo el servidor:***./catalina.sh run***

Anexo V. Especificación del código de la parte servidora

En este anexo se describe de forma completa la capa de control del sistema, donde veremos el modelo de negocio seguido a la hora de implementar las funcionalidades del sistema.

A continuación se muestra el diagrama de clases diseñado de forma detallada.

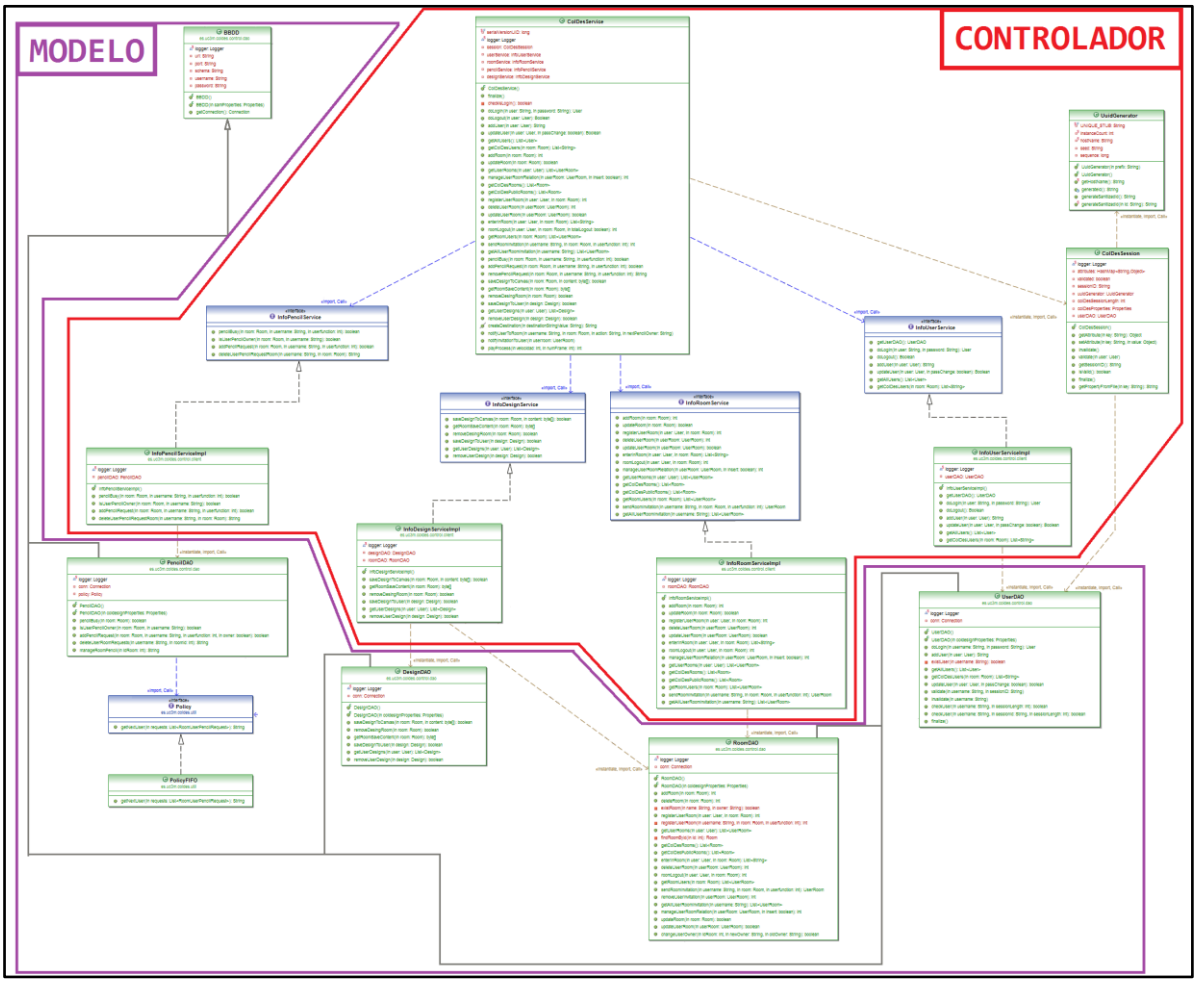


Figura 119: Diagrama de clases general

Antes de explicar el diagrama de clases representado en la figura anterior, es necesario hacer una introducción de las distintas clases creadas para modelar los distintos elementos del sistema, para de esta forma tener una mejor noción sobre el tipo de elementos que devuelven y reciben las distintas funciones del sistema.

El modelado de estos elementos se ha realizado mediante la creación de distintos **POJOS** que hacen referencia a cada uno de los elementos del sistema. **POJO** es un acrónimo de *Plain Old Java Objects*. El nombre se usa para enfatizar que el objeto dado es un objeto Java normal, no un objeto especial. En nuestro caso, en estas clases

únicamente se definen los atributos y los métodos para poder accederlos y modificarlos (get y set).

En los puntos que siguen describiremos cada uno de los POJOS definidos para el sistema, centrándonos en la descripción de cada uno de los atributos que hay definidos en cada uno de ellos.

Clase User

Mediante esta clase modelamos lo que sería un usuario.

Nombre	User.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	name	*String*	Nombre real del usuario.	
	surname1	*String*	Primer apellido del usuario.	
	surname2	*String*	Segundo apellido del usuario.	
	email	*String*	Correo electrónico del usuario.	
	username	*String*	Nombre de usuario del usuario.	
	password	*String*	Contraseña del usuario.	
	admin	*Boolean*	Booleano que indica si el usuario es administrador o no.	
	designer	*Boolean*	Booleano que indica si el usuario es diseñador o no.	
	active	*Boolean*	Booleano que indica si el usuario esta activo o no.	

Tabla 80: Atributos POJO User

Clase Room

Mediante esta clase modelamos lo que sería una sala para el sistema.

Nombre	Room.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	id	*int*	Id de la sala.	
	name	*String*	Nombre de la sala.	
	description	*String*	Descripción de la sala.	
	owner	*String*	Propietario de la sala. Por defecto siempre el creador de la misma.	
	privateRoom	*Boolean*	Booleano que indica si la sala es privada o no.	
	participationType	*int*	Tipo de participación que soporta la sala.	
	status	*int*	Estado de la sala: abierta o cerrada.	
	creationDate	*Date*	Fecha de creación de la sala.	
	modificationDate	*Date*	Fecha de última modificación de la sala.	

Tabla 81: Atributos POJO Room

Clase UserRoom

Esta clase pretende modelar la relación entre salas y usuarios. Estas relaciones lo que indican es que usuarios están presentes en que salas.

Nombre	RoomUser.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	room	*Room*	Sala a la que hace referencia la relación.	
	roomName	*String*	Nombre de la sala.	
	ownerUserName	*String*	Nombre del propietario de la sala.	
	userName	*String*	Nombre de usuario del usuario de la relación.	
	userFunction	*int*	Función que desempeña el usuario de la relación dentro de la sala.	
	userFunctionDescription	*String*	Descripción de la función del usuario dentro de la sala.	

Tabla 82: Atributos POJO RoomUser

Clase RoomUserPencilRequest

Esta clase modela una petición de un pincel por parte de un usuario en una determinada sala.

Nombre	RoomUserPencilRequest.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	idRoom	*int*	Id de la sala donde se realiza la petición del pincel.	
	userName	*String*	Nombre de usuario del usuario que hace la petición del pincel dentro de la sala.	
	userFunction	*int*	Función que desempeña dentro de la sala el usuario que realiza la petición.	
	requestTime	*Date*	Momento en el que se realiza la petición del pincel.	
	pencilOwner	*Boolean*	Booleano que indica si esta petición es la actual propietaria del pincel dentro de la sala.	

Tabla 83: Atributos POJO RoomUserPencilRequest

Clase Design

Esta clase modela lo que es un diseño dentro del sistema. Este diseño en el modelo actual es el que se usa para representar los diseños almacenados por los usuarios dentro del sistema. El diseño asociado a la sala se almacena directamente como un array de bytes en base de datos.

Nombre	Design.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	username	*String*	Nombre del usuario que guarda el diseño.	
	designcontent	*byte[]*	Contenido del diseño.	
	designname	*String*	Nombre asignado por el usuario para el diseño.	

Tabla 84: Atributos POJO Design

Una vez explicadas las clases de los elementos del sistema, podemos pasar a explicar de forma detallada el diagrama presentado en la Figura 24. Vamos a ir analizándolo elemento a elemento, explicando mediante una tabla todas las funciones y atributos definidos en cada una de las clases que conforman la parte servidora de la aplicación.

Controlador

La parte controladora tiene cinco grandes bloques. Por un lado tenemos la clase que sirve de enlace entre la vista y el controlador del sistema, y por otro lado tenemos los cuatro módulos que la herramienta ha de controlar: usuarios, salas, pinceles y diseños.

Primero empezaremos por la clase que sirve de enlace entre la vista y el controlador dentro de sistema.

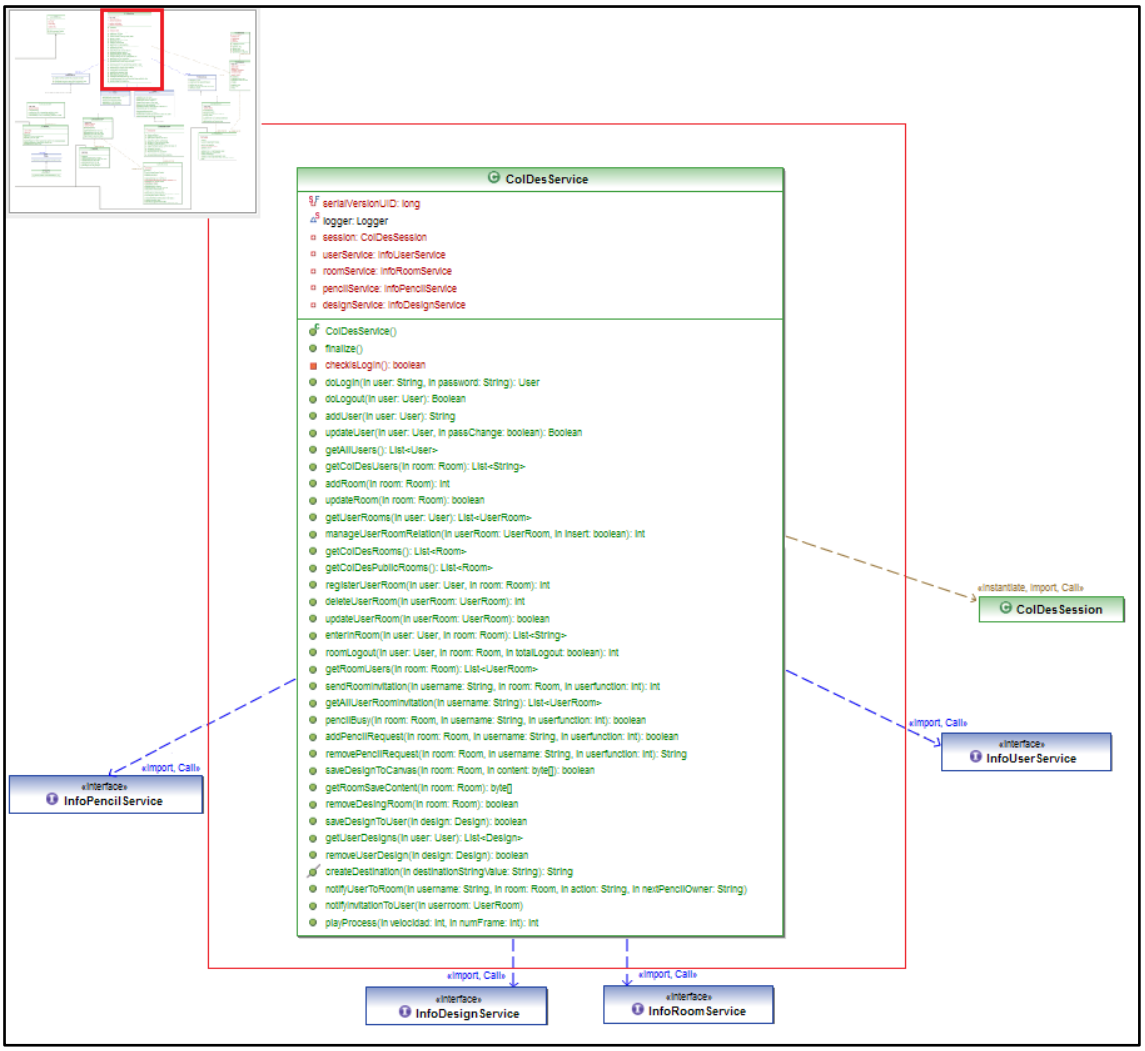


Figura 120: ColDesService

Como hemos comentado esta clase sirve como enlace entre la parte cliente y el servidor. En esta clase están definidos todos los métodos que la vista puede necesitar para ofrecer al usuario los servicios que la aplicación ha de satisfacer. Cada uno de estos métodos hará uso de un *service* distinto dependiendo de a que modulo haga referencia dicha función.

Nombre	ColDesService.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	session	*ColDesSession*	Variable que gestiona las sesiones del sistema	
	userService	*InfoUserService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con los usuarios	
	roomService	*InfoRoomService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con las salas	
	pencilService	*InfoPencilService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con los diseños del sistema, tanto de usuarios como de salas.	
	designService	*InfoDesignService*	Service mediante el cual se permite invocar todas las funcionalidades relacionadas con los diseños del sistema, tanto de usuarios como de salas	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	ColDesService	*Constructor*	-	
	Descripción			
	Constructor por defecto encargado de inicializar la sesión y los distintos service del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	finalize	*void*	-	
	Descripción			
	Función encargada de finalizar la sesión del DAO del usuario.			
	Nombre	*Tipo*	*Argumentos*	
	checkIsLogIn	*Boolean*	-	
	Descripción			
	Función que verifica si el usuario aún tiene una sesión valida dentro del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	doLogin	*User*	*String* user *String* password	
	Descripción			
	Función encargada de realizar la autenticación del usuario dentro del sistema. Para ello recurre al Service de usuarios.			
	Nombre	*Tipo*	*Argumentos*	
	doLogout	*Boolean*	*User* user	

	Descripción		
	Función encargada de finalizar la sesión del usuario autenticado en el sistema. Para ello invalida la sesión, y usa el Service de usuarios para finalizar las acciones pendientes del usuario en el sistema.		
	Nombre	*Tipo*	*Argumentos*
	addUser	*String*	*User* user
	Descripción		
	Añade un nuevo usuario al sistema mediante el Service de usuarios.		
	Nombre	*Tipo*	*Argumentos*
	updateUser	*Boolean*	*User* user *boolean* passChange
	Descripción		
	Actualiza los datos de un usuario mediante el Service de usuarios. En caso de cambio de contraseña se pasa a true el *boolean* passChange para indicar que hay que cambiar la contraseña del usuario.		
	Nombre	*Tipo*	*Argumentos*
	getAllUsers	*List <User>*	-
	Descripción		
	Función que obtiene todos los usuarios del sistema mediante el Service de usuarios.		
	Nombre	*Tipo*	*Argumentos*
	getColDesUsers	*List <String>*	*Room* room
	Descripción		
	Función que devuelve todos los usuarios que están en una determinada sala en el momento actual.		
	Nombre	*Tipo*	*Argumentos*
	addRoom	*int*	*Room* room
	Descripción		
	Añade una nueva sala al sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	updateRoom	*boolean*	*Room* room
	Descripción		
	Actualiza los datos referentes de una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getUserRooms	*List <UserRoom>*	*User* user
	Descripción		
	Obtiene la lista de salas en las cuales está participando un usuario, así como la función que desempeña en cada una de las salas.		
	Nombre	*Tipo*	*Argumentos*
	manageUserRoomRelation	*int*	*UserRoom* userRoom *boolean* insert
	Descripción		
	Función encargada de gestionar las relaciones de los usuarios con las distintas salas del sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getColDesRooms	*List <Room>*	-
	Descripción		
	Obtiene las distintas salas del sistema sin excepción mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getColDesPublicRooms	*List <Room>*	-
	Descripción		

	Obtiene las salas públicas del sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	registerUserRoom	*int*	*User* user *Room* room
	Descripción		
	Asocia un usuario a una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	deleteUserRoom	*int*	*UserRoom* userRoom
	Descripción		
	Borrar un usuario de una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	updateUserRoom	*Boolean*	*UserRoom* userRoom
	Descripción		
	Actualiza una relación usuario - sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	enterInRoom	*List <String>*	*User* user *Room* room
	Descripción		
	Función mediante la cual se notifica al sistema la entrada de un usuario a una sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	roomLogout	*int*	*User* user *Room* room *Boolean* totalLogout
	Descripción		
	Función que notifica al sistema cuando un usuario sale de la sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getRoomUsers	*List <UserRoom>*	*Room* room
	Descripción		
	Obtiene las distintas relaciones que tiene una sala con los usuarios del sistema mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	sendRoomInvitation	*int*	*String* username *Room* room *int* userfunction
	Descripción		
	Función mediante la cual se le envía una invitación a sala mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	getAllUserroomInvitation	*List <UserRoom>*	*String* username
	Descripción		
	Función mediante la cual se obtienen todas las invitaciones que hay en el sistema de un usuario mediante el Service de salas.		
	Nombre	*Tipo*	*Argumentos*
	pencilBusy	*Boolean*	*Room* room *String* username *int* userfunction
	Descripción		
	Función que verifica si el pincel está siendo usado en una sala mediante el Service de pinceles.		

Nombre	*Tipo*	*Argumentos*
addPencilRequest	*Boolean*	*Room* room *String* username *int* userfunction
Descripción		
Función mediante la cual un usuario realiza una petición para poder usar el pincel mediante el Service de pinceles. Adicionalmente se manda un mensaje a todos los usuarios de la sala para notificar la del pincel.		
Nombre	*Tipo*	*Argumentos*
removePencilRequest	*String*	*Room* room *String* username *int* userfunction
Descripción		
Función mediante la cual se elimina una petición de un pincel de una sala mediante el Service de pinceles.		
Nombre	*Tipo*	*Argumentos*
saveDesignToCanvas	*Boolean*	*Room* room *Byte[]* content
Descripción		
Función encargada de asociar un diseño a una sala mediante el Service de diseños.		
Nombre	*Tipo*	*Argumentos*
getRoomSaveContent	*byte[]*	*Room* room
Descripción		
Obtiene el diseño asociado a una sala mediante el Service de diseños.		
Nombre	*Tipo*	*Argumentos*
removeDesingRoom	*Boolean*	*Room* room
Descripción		
Función encargada de borrar un diseño de una sala mediante el Service de diseños.		
Nombre	*Tipo*	*Argumentos*
saveDesignToUser	*Boolean*	*Design* design
Descripción		
Guarda para un usuario un diseño mediante el Service de diseños.		
Nombre	*Tipo*	*Argumentos*
getUserDesigns	*List <Design>*	*User* user
Descripción		
Función que devuelve los distintos diseños asociados a un usuario mediante el Service de diseños.		
Nombre	*Tipo*	*Argumentos*
removeUserDesign	*Boolean*	*Design* design
Descripción		
Borra un diseño asociado a un usuario mediante el Service de diseños.		
Nombre	*Tipo*	*Argumentos*
createDestination	*String*	*String* destinationStringValue
Descripción		
Crea canales que usara la aplicación para sincronizar a los distintos usuarios mediante los chats y lienzos de la sala. Este método en la última versión esta *deprecated*.		
Nombre	*Tipo*	*Argumentos*

	notifyUserToRoom	*void*	*String* username *Room* room *String* action *String* nextPencilOwner
	Descripción		
	Función mediante la cual se notifica de distintas acciones en una sala al resto de usuarios de la sala en cuestión.		
	Nombre	*Tipo*	*Argumentos*
	notifyInvitationToUser	*void*	*UserRoom userroom*
	Descripción		
	Función mediante la cual se notifica al usuario en tiempo real de la invitación a una sala.		
	Nombre	*Tipo*	*Argumentos*
	playProcess	*int*	*int* velocidad *int* numFrame
	Descripción		
	Función encargada de simular el *play* de acciones de la sala.		

Tabla 85: Clase ColDesService

El siguiente diagrama contiene las clases que se encargan de controlar las sesiones de los usuarios dentro del sistema: *ColDesSession* y *UuidGeneratorr*. Esta segunda clase únicamente se encarga de generar un id de sesión basándose en el host del usuario que inicia sesión y en el momento en el cual inicia dicha sesión.

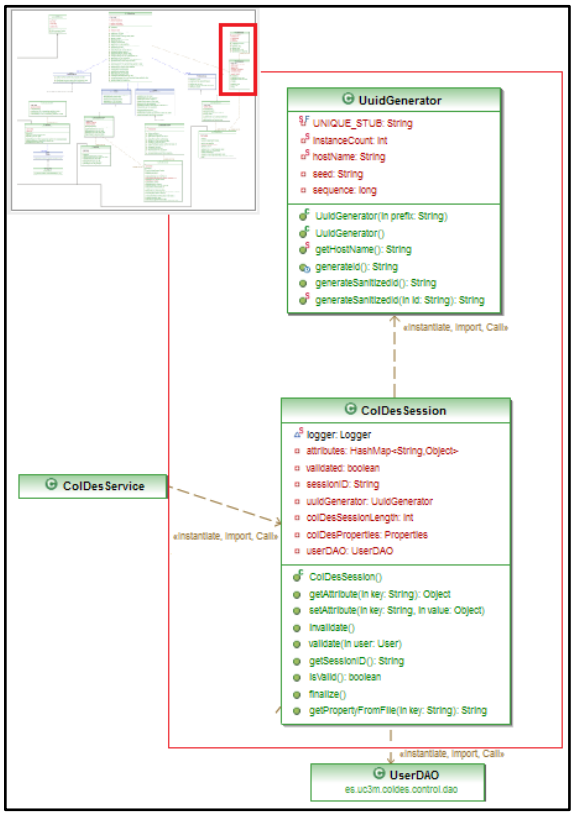


Figura 121: ColDesSession - UuidGenerator

Nombre	ColDesSession.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	attributes	*HashMap <String, Object>*	Mapa de atributos de una sesión.	
	validated	*Boolean*	Flag que indica si la sesión esta validada o no.	
	sessionID	*String*	String que modela el ID de sesión único para este usuario.	
	uuidGenerator	*UuidGenerator*	Generador de ids de sesión.	
	colDesSessionLength	*int*	Tiempo de validez de la sesión dentro del sistema en minutos.	
	colDesProperties	*Properties*	Propiedades del sistema cargadas del fichero de propiedades de la aplicación.	
	userDAO	*UserDAO*	DAO encargado de aportar a la sesión los datos necesarios de los usuarios del sistema.	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	ColDesSession	*Constructor*	-	
	Descripción			
	Constructor por defecto que crea un nuevo objeto de sesión, inicialmente invalidada.			
	Nombre	*Tipo*	*Argumentos*	
	getAttribute	*Object*	*String* key	
	Descripción			
	Lee un atributo de sesión si esta está validada.			
	Nombre	*Tipo*	*Argumentos*	
	setAttribute	*void*	*String* key *Object* value	
	Descripción			
	Guarda un nuevo atributo en sesión, o lo modifica si ya existía, siempre que la sesión esté validada. Si la sesión esta invalidada, ignora la petición.			
	Nombre	*Tipo*	*Argumentos*	
	invalidate	*void*	-	
	Descripción			
	Invalida una sesión existente, sea cual sea el motivo.			
	Nombre	*Tipo*	*Argumentos*	
	validate	*void*	*User* user	
	Descripción			
	Valida una sesión existente como consecuencia de un inicio de sesión, asociada al usuario que se recibe como argumento. Este objeto, conteniendo toda la información relativa al usuario, es almacenada como atributo de sesión.			
	Nombre	*Tipo*	*Argumentos*	
	getSessionID	*String*	-	
	Descripción			
	Devuelve el valor de la sesión.			
	Nombre	*Tipo*	*Argumentos*	
	isValid	*Boolean*	-	
	Descripción			

<table>
<tr><td></td><td colspan="3">Comprueba si la sesión del usurio existe y, en ese caso, que no haya caducado. Para comprobar la caducidad usa la propiedad "coldesSessionLength", que indica el tiempo en minutos que puede estar la sesión inactiva antes de que caduque. Para que una sesión sea válida, deben cumplirse las siguientes tres condiciones a la vez:
El atributo sessionID no es null
El valor del atributo sessionID es igual al valor almacenado en BBDD para este user
El tiempo transcurrido desde el último movimiento hasta el momento actual es menor indicado por la propiedad colDesSessionLength
Si se cumplen todas las condiciones, se actualizará en BBDD la fecha y hora del último movimiento usando la fecha y hora actual, y se devolverá true. Si alguna de las condiciones falla, se eliminará el valor sessionID almancenado en BBDD y la fecha del último movimiento, si es que existen, y se devolverá false.</td></tr>
<tr><td></td><td>Nombre</td><td>Tipo</td><td>Argumentos</td></tr>
<tr><td></td><td>finalize</td><td>void</td><td>-</td></tr>
<tr><td></td><td colspan="3">Descripción</td></tr>
<tr><td></td><td colspan="3">Finaliza el DAO userDAO para asegurarnos de que se cierran las conexiones con base de datos de forma correcta.</td></tr>
<tr><td></td><td>Nombre</td><td>Tipo</td><td>Argumentos</td></tr>
<tr><td></td><td>getPropertyFromFile</td><td>String</td><td>String key</td></tr>
<tr><td></td><td colspan="3">Descripción</td></tr>
<tr><td></td><td colspan="3">Consulta una propiedad del archivo de propiedades, si la sesión es válida, y si la encuentra la devuelve.</td></tr>
</table>

Tabla 86: Clase ColDesSession

A continuación explicaremos a fondo cada uno de los *service* presentes en el proyecto: de usuarios, de salas, de diseños y de pinceles.

- Service de usuarios

El *service* de usuarios es el encargado de gestionar todas las peticiones que tienen que ver con el módulo de usuarios del sistema.

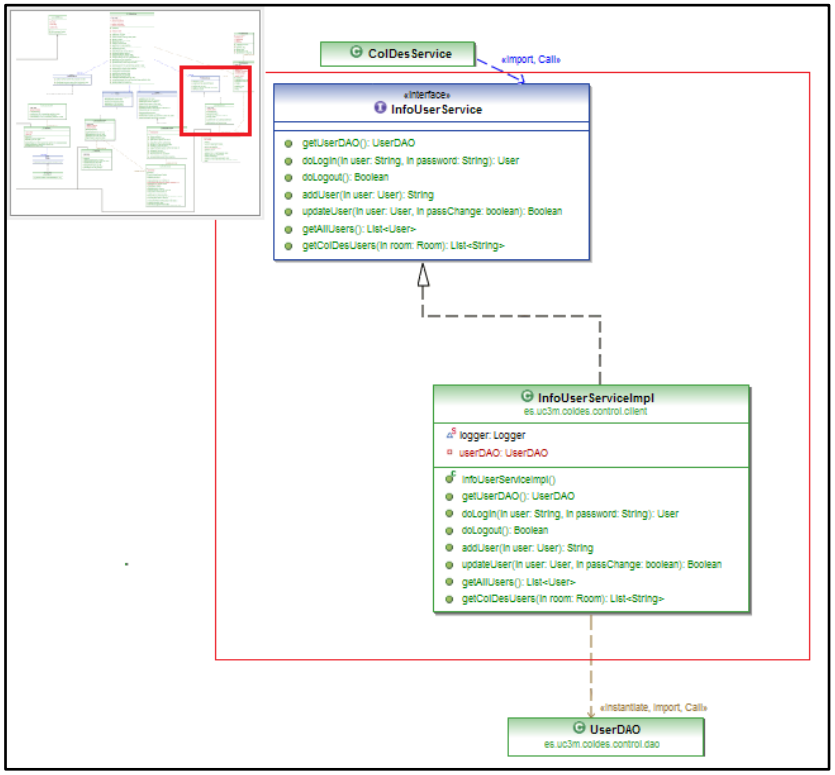


Figura 122: Service de Usuarios

Este *service* es usado desde la capa más externa del controlador del sistema, desde el **ColDesService**, y se basa por otro lado en la parte del modelo encargada de gestionar los usuarios dentro de la base de datos mediante la clase **UserDAO** (las clases del modelo las veremos más adelante).

Comenzaremos explicando la interfaz que define el *service* de usuario, y la clase que posteriormente implementa esta interfaz.

Dado que los métodos y su fin es el mismo en la interfaz que en la clase que los implementa, en la tabla de la interfaz nos limitaremos a mostrar la nomenclatura de los métodos que esta define y en la tabla de la clase que implementa la interfaz describiremos detalladamente su funcionalidad.

Nombre	InfoUserService.java		**Tipo**	Interfaz
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	getUserDAO	*UserDAO*	-	
	Nombre	*Tipo*	*Argumentos*	
	doLogin	*User*	*String* user *String* password	
	Nombre	*Tipo*	*Argumentos*	
	doLogout	*Boolean*	*User* user	
	Nombre	*Tipo*	*Argumentos*	
	addUser	*String*	*User* user	
	Nombre	*Tipo*	*Argumentos*	
	updateUser	*Boolean*	*User* user *boolean* passChange	
	Nombre	*Tipo*	*Argumentos*	
	getAllUsers	*List <User>*	-	
	Nombre	*Tipo*	*Argumentos*	
	getColDesUsers	*List <String>*	*Room* room	

Tabla 87: Interfaz InfoUserService

Nombre	InfoUserServiceImpl.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	userDAO	*UserDAO*	DAO encargado de la parte de Usuario del sistema	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	InfoUserServiceImpl	*Constructor*	-	
	Descripción			
	Constructor por defecto encargado de inicializar el DAO de Usuarios.			
	Nombre	*Tipo*	*Argumentos*	
	doLogin	*User*	*String* user *String* password	
	Descripción			
	Función encargada de realizar la autenticación del usuario dentro del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	doLogout	*Boolean*	*User* user	
	Descripción			
	Función encargada de finalizar la sesión del usuario autenticado en el sistema. Para ello invalida la sesión, y finaliza las acciones pendientes del usuario en el sistema.			
	Nombre	*Tipo*	*Argumentos*	
	addUser	*String*	*User* user	
	Descripción			
	Añade un nuevo usuario al sistema.			
	Nombre	*Tipo*	*Argumentos*	
	updateUser	*Boolean*	*User* user *boolean* passChange	
	Descripción			
	Actualiza los datos de un usuario. En caso de cambio de contraseña se pasa a true el *boolean* passChange para indicar que hay que cambiar la contraseña del usuario.			
	Nombre	*Tipo*	*Argumentos*	

	getAllUsers	*List <User>*	-
	Descripción		
	Función que obtiene todos los usuarios del sistema.		
	Nombre	*Tipo*	*Argumentos*
	getColDesUsers	*List <String>*	*Room* room
	Descripción		
	Función que devuelve todos los usuarios que están en una determinada sala en el momento actual.		

Tabla 88: Clase InfoUserServiceImpl

- Service de salas

El *service* de salas es el encargado de gestionar todas las peticiones que tienen que ver con el módulo de salas del sistema.

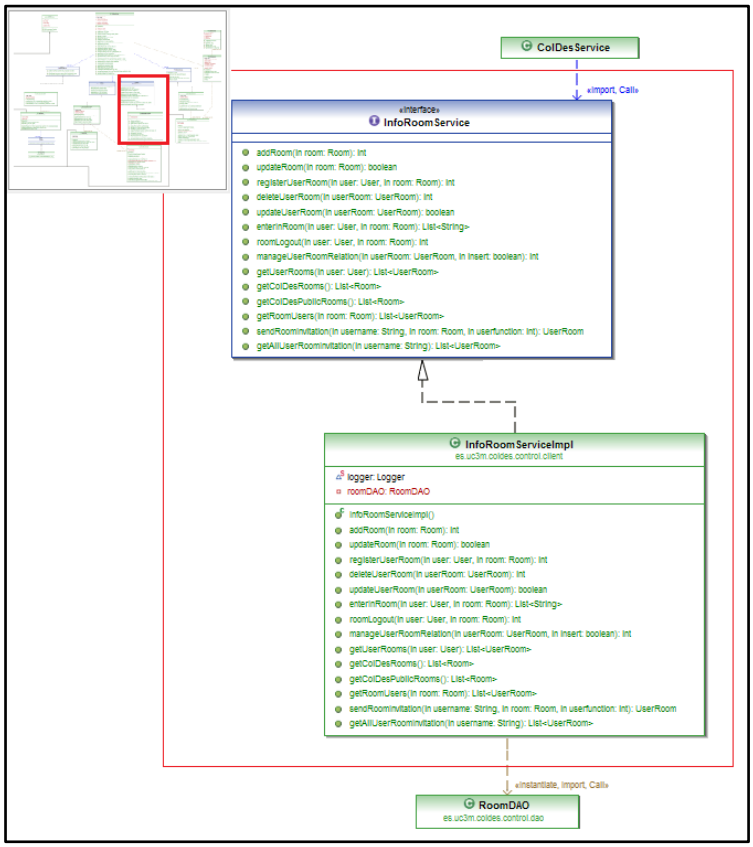


Figura 123: Service de Salas

Este *service* es usado desde la capa más externa del controlador del sistema, desde el **ColDesService**, y se basa por otro lado en la parte del modelo encargado de gestionar las salas, las relaciones con los usuarios y las invitaciones a salas dentro de la base de datos mediante la clase **RoomDAO**.

Comenzaremos explicando la interfaz que define el *service* de salas, y la clase que posteriormente implementa esta interfaz.

Dado que los métodos y su fin es el mismo en la interfaz que en la clase que los implementa, en la tabla de la interfaz nos limitaremos a mostrar la nomenclatura de los métodos que esta define y en la tabla de la clase que implementa la interfaz describiremos detalladamente su funcionalidad.

Nombre	InfoRoomService.java		**Tipo**	Interfaz
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	addRoom	*int*	*Room* room	
	Nombre	*Tipo*	*Argumentos*	
	updateRoom	*boolean*	*Room* room	
	Nombre	*Tipo*	*Argumentos*	
	registerUserRoom	*int*	*User* user *Room* room	
	Nombre	*Tipo*	*Argumentos*	
	deleteUserRoom	*int*	*UserRoom* userRoom	
	Nombre	*Tipo*	*Argumentos*	
	updateUserRoom	*Boolean*	*UserRoom* userRoom	
	Nombre	*Tipo*	*Argumentos*	
	enterInRoom	*List <String>*	*User* user *Room* room	
	Nombre	*Tipo*	*Argumentos*	
	roomLogout	*int*	*User* user *Room* room *Boolean* totalLogout	
	Nombre	*Tipo*	*Argumentos*	
	manageUserRoomRelation	*int*	*UserRoom* userRoom *boolean* insert	
	Nombre	*Tipo*	*Argumentos*	
	getUserRooms	*List <UserRoom>*	*User* user	
	Nombre	*Tipo*	*Argumentos*	
	getColDesRooms	*List <Room>*	-	
	Nombre	*Tipo*	*Argumentos*	
	getColDesPublicRooms	*List <Room>*	-	
	Nombre	*Tipo*	*Argumentos*	
	getRoomUsers	*List <UserRoom>*	*Room* room	
	Nombre	*Tipo*	*Argumentos*	
	sendRoomInvitation	*int*	*String* username *Room* room *int* userfunction	
	Nombre	*Tipo*	*Argumentos*	
	getAllUserroomInvitation	*List <UserRoom>*	*String* username	

Tabla 89: Interfaz InfoRoomService

Nombre	InfoRoomServiceImpl.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	userDAO	*RoomDAO*	DAO encargado de la parte de Salas del sistema	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	InfoRoomServiceImpl	*Constructor*	-	
	Descripción			
	Constructor por defecto encargado de inicializar el DAO de Salas.			
	Nombre	*Tipo*	*Argumentos*	
	addRoom	*int*	*Room* room	
	Descripción			
	Añade una nueva sala al sistema.			
	Nombre	*Tipo*	*Argumentos*	
	updateRoom	*boolean*	*Room* room	
	Descripción			
	Actualiza los datos referentes de una sala del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	registerUserRoom	*int*	*User* user *Room* room	
	Descripción			
	Asocia un usuario a una sala.			
	Nombre	*Tipo*	*Argumentos*	
	deleteUserRoom	*int*	*UserRoom* userRoom	
	Descripción			
	Actualiza una relación usuario sala. En caso de que el usuario sea el propietario de la sala se borra también la sala.			
	Nombre	*Tipo*	*Argumentos*	
	updateUserRoom	*Boolean*	*UserRoom* userRoom	
	Descripción			
	Actualiza una relación usuario - sala.			
	Nombre	*Tipo*	*Argumentos*	
	enterInRoom	*List <String>*	*User* user *Room* room	
	Descripción			
	Función mediante la cual se notifica al sistema la entrada de un usuario a una sala.			
	Nombre	*Tipo*	*Argumentos*	
	roomLogout	*int*	*User* user *Room* room *Boolean* totalLogout	
	Descripción			
	Función que notifica al sistema cuando un usuario sale de la sala.			
	Nombre	*Tipo*	*Argumentos*	
	manageUserRoomRelation	*int*	*UserRoom* userRoom *boolean* insert	
	Descripción			
	Función encargada de gestionar las relaciones de los usuarios con las distintas salas del sistema.			
	Nombre	*Tipo*	*Argumentos*	
	getUserRooms	*List <UserRoom>*	*User* user	
	Descripción			

	Obtiene la lista de salas en las cuales está participando un usuario, así como la función que desempeña en cada una de las salas.		
	Nombre	*Tipo*	*Argumentos*
	getColDesRooms	*List <Room>*	-
	Descripción		
	Obtiene las distintas salas del sistema sin excepción.		
	Nombre	*Tipo*	*Argumentos*
	getColDesPublicRooms	*List <Room>*	-
	Descripción		
	Obtiene las salas públicas del sistema.		
	Nombre	*Tipo*	*Argumentos*
	getRoomUsers	*List <UserRoom>*	*Room* room
	Descripción		
	Obtiene las distintas relaciones que tiene una sala con los usuarios del sistema.		
	Nombre	*Tipo*	*Argumentos*
	sendRoomInvitation	*int*	*String* username *Room* room *int* userfunction
	Descripción		
	Función mediante la cual se le envía una invitación a sala a un determinado usuario.		
	Nombre	*Tipo*	*Argumentos*
	getAllUserroomInvitation	*List <UserRoom>*	*String* username
	Descripción		
	Función mediante la cual se obtienen todas las invitaciones que hay en el sistema de un usuario.		

Tabla 90: Clase InfoRoomServiceImpl

-

- Service de pinceles

El *service* de pinceles es el encargado de gestionar los pinceles de las salas en las cuales la participación se realiza por turnos. Este *service* gestiona todas las peticiones de pinceles de los usuarios, notifica las peticiones al resto de usuarios de las salas y controla si el pincel tiene o no propietario.

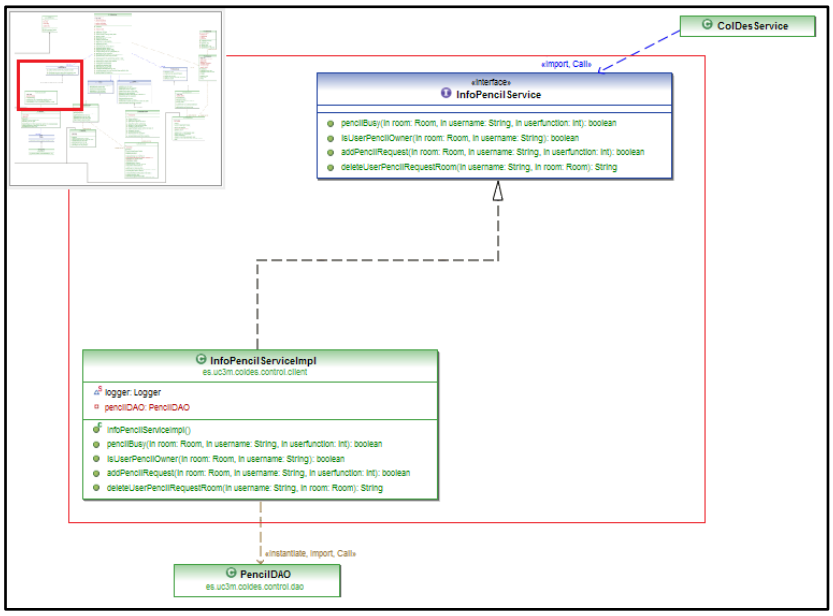


Figura 124: Service de Pinceles

Este *service* es usado desde la capa más externa del controlador del sistema, desde el **ColDesService**, y se basa por otro lado en la parte del modelo encargado de gestionar los pinceles dentro de la base de datos mediante la clase **PencilDAO**.

Comenzaremos explicando la interfaz que define el *service* de pinceles, y la clase que posteriormente implementa esta interfaz.

Dado que los métodos y su fin es el mismo en la interfaz que en la clase que los implementa, en la tabla de la interfaz nos limitaremos a mostrar la nomenclatura de los métodos que esta define y en la tabla de la clase que implementa la interfaz describiremos detalladamente su funcionalidad.

Nombre	InfoPencilService.java		**Tipo**	Interfaz
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	pencilBusy	*Boolean*	*Room* room *String* username *int* userfunction	
	Nombre	*Tipo*	*Argumentos*	
	isUserPencilOwner	*Boolean*	*Room* room *String* username	
	Nombre	*Tipo*	*Argumentos*	

	addPencilRequest	*Boolean*	*Room* room *String* username *int* userfunction
	Nombre	*Tipo*	*Argumentos*
	removePencilRequest	*String*	*Room* room *String* username *int* userfunction

Tabla 91: Interfaz InfoPencilService

Nombre	InfoPencilServiceImpl.java			**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*		
	pencilDAO	*PencilDAO*	DAO encargado de la parte de Pinceles del sistema		
Métodos	*Nombre*	*Tipo*	*Argumentos*		
	InfoPencilServiceImpl	*Constructor*	-		
	Descripción				
	Constructor por defecto encargado de inicializar el DAO de Pinceles.				
	Nombre	*Tipo*	*Argumentos*		
	pencilBusy	*Boolean*	*Room* room *String* username *int* userfunction		
	Descripción				
	Función que verifica si el pincel está siendo usado en una sala, en caso de que el pincel no tenga un dueño en el preciso momento, el usuario que ha entrado se convierte en el actual dueño del pincel.				
	Nombre	*Tipo*	*Argumentos*		
	isUserPencilOwner	*Boolean*	*Room* room *String* username		
	Descripción				
	Función que indica si un usuario dado es el dueño del pincel en una determinada sala.				
	Nombre	*Tipo*	*Argumentos*		
	addPencilRequest	*Boolean*	*Room* room *String* username *int* userfunction		
	Descripción				
	Función mediante la cual un usuario realiza una petición para poder usar el pincel. Primero se verifica si el pincel de la sala tiene ya un propietario, de no ser así, la nueva petición de pincel se convierte automáticamente en la dueña del pincel.				
	Nombre	*Tipo*	*Argumentos*		
	removePencilRequest	*String*	*Room* room *String* username *int* userfunction		
	Descripción				
	Función mediante la cual se elimina una petición de un pincel de una sala.				

Tabla 92: Clase InfoPencilServiceImpl

- Service de diseños

El *service* de diseños es el encargado de gestionar los diseños del sistema. Se encarga de asociar diseños a salas y a usuarios. Una sala únicamente puede tener asociado en todo momento un diseño. Por otro lado, un usuario a la hora de guardarse un diseño ha de especificar un nombre para dicho diseño, y este puede almacenar tantos diseños como desee.

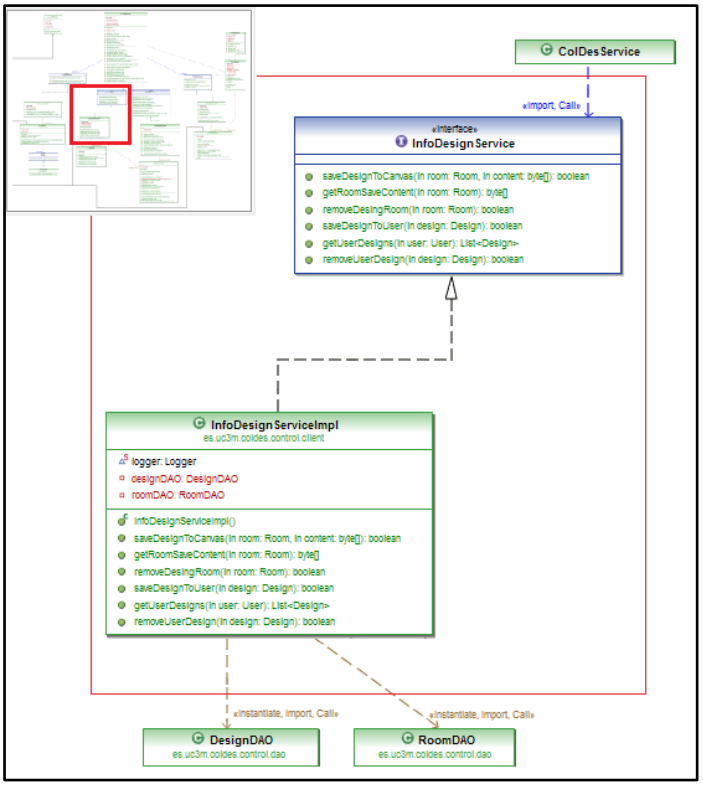


Figura 125: Service de Diseños

Este *service* es usado desde la capa más externa del controlador del sistema, desde el **ColDesService**, y se basa por otro lado en la parte del modelo encargado de gestionar los diseños dentro de la base de datos mediante la clase **DesignDAO**. También se basa en el DAO (**RoomDAO**) de salas para poder asociar a una sala un diseño dado.

Comenzaremos explicando la interfaz que define el *service* de diseños, y la clase que posteriormente implementa esta interfaz.

Dado que los métodos y su fin es el mismo en la interfaz que en la clase que los implementa, en la tabla de la interfaz nos limitaremos a mostrar la nomenclatura de los métodos que esta define y en la tabla de la clase que implementa la interfaz describiremos detalladamente su funcionalidad.

Nombre	InfoDesignService.java		**Tipo**	Interfaz
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	saveDesignToCanvas	*Boolean*	*Room* room *Byte[]* content	
	Nombre	*Tipo*	*Argumentos*	
	getRoomSaveContent	*byte[]*	*Room* room	
	Nombre	*Tipo*	*Argumentos*	
	removeDesingRoom	*Boolean*	*Room* room	
	Nombre	*Tipo*	*Argumentos*	
	saveDesignToUser	*Boolean*	*Design* design	
	Nombre	*Tipo*	*Argumentos*	
	getUserDesigns	*List <Design>*	*User* user	
	Nombre	*Tipo*	*Argumentos*	
	removeUserDesign	*Boolean*	*Design* design	

Tabla 93: Interfaz InfoDesignService

Nombre	InfoPencilServiceImpl.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	designDAO	*DesignDAO*	DAO encargado de la parte de Diseños del sistema	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	InfoDesignServiceImpl	*Constructor*	-	
	Descripción			
	Constructor por defecto encargado de inicializar el DAO de Diseños.			
	Nombre	*Tipo*	*Argumentos*	
	saveDesignToCanvas	*Boolean*	*Room* room *Byte[]* content	
	Descripción			
	Función encargada de asociar un diseño a una sala. Primero se asocia el diseño a la sala y posteriormente se actualizan los datos de la sala.			
	Nombre	*Tipo*	*Argumentos*	
	getRoomSaveContent	*byte[]*	*Room* room	
	Descripción			
	Obtiene el diseño asociado a una sala.			
	Nombre	*Tipo*	*Argumentos*	
	removeDesingRoom	*Boolean*	*Room* room	
	Descripción			
	Función encargada de borrar un diseño de una sala.			
	Nombre	*Tipo*	*Argumentos*	
	saveDesignToUser	*Boolean*	*Design* design	
	Descripción			
	Guarda para un usuario un diseño.			
	Nombre	*Tipo*	*Argumentos*	
	getUserDesigns	*List <Design>*	*User* user	
	Descripción			

	Función que devuelve los distintos diseños asociados a un usuario.		
	Nombre	*Tipo*	*Argumentos*
	removeUserDesign	*Boolean*	*Design* design
	Descripción		
	Borra un diseño asociado a un usuario.		

Tabla 94: Clase InfoDesignServiceImpl

Modelo

El modelo del sistema se centra en los cuatro grandes módulos que el sistema ha de gestionar y controlar: usuarios, salas, pinceles y diseños.

Las clases diseñadas para llevar este control heredan todas de una superclase que define los métodos básicos de conexión en base de datos y de carga de parámetros mediante un fichero de configuración:

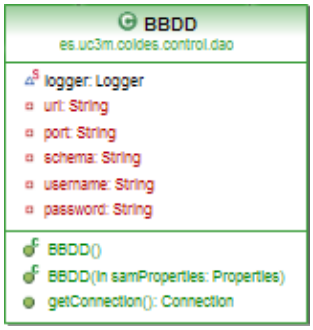


Figura 126: Clase padre del modelo

Nombre	BBDD.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	url	*String*	URL de la base de datos.	
	port	*String*	Puerto en el que se encuentra escuchando la base de datos.	
	schema	*String*	Schema que contiene la estructura de tablas de la aplicación.	
	username	*String*	Nombre de usuario para conectarse a la base de datos.	
	password	*String*	Contraseña del usuario de la base de datos.	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	BBDD	*Constructor*	-	
	Descripción			
	Constructor por defecto en el que se leen las propiedades de la base de datos del fichero de propiedades *coldes.properties.*			
	Nombre	*Tipo*	*Argumentos*	
	BBDD	*Contructor*	*Properties* ColDesProperties	
	Descripción			
	Constructor que recibe ya las distintas propiedades de la conexión a base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	getConnection	*Connection*	-	
	Descripción			
	Función mediante la cual se obtiene una conexión a la base de datos del sistema.			

Tabla 95: Clase BBDD

El fichero de configuración que usa esta clase para cargar los datos referentes a la base de datos se denomina en el sistema *coldes.properties* y sigue la siguiente estructura:

```
####################################
# Archivo de propiedades de ColDes #
####################################

# Duración de la sesión, medido en minutos
session.length=30

# URL de la máquina
server.url=localhost

# Acceso a la base de datos
db.url=localhost
db.port=3306
db.schema=coldesbd
db.username=coldesbd
db.password=coldesbd
```

De forma general, los distintos **DAO's** (*Data Access Object*) definidos en el sistema seguirían la siguiente estructura de clases:

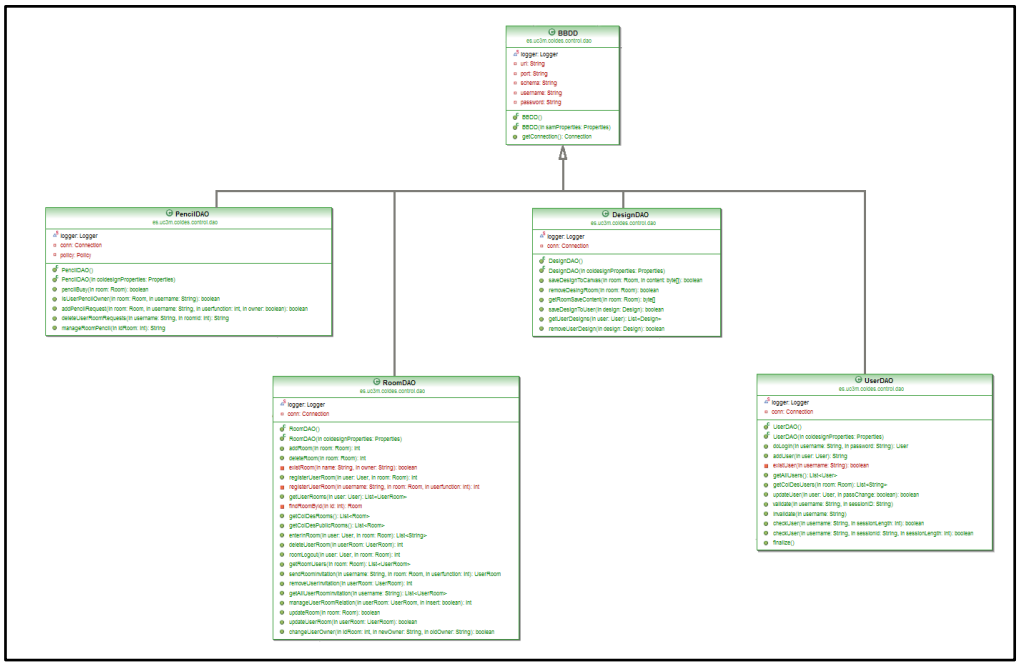


Figura 127: Estructura de clases del modelo

A continuación iremos viendo uno a uno las clases creadas para gestionar la parte del modelo de los elementos del sistema: usuarios, salas, diseños y pinceles.

- DAO de usuarios

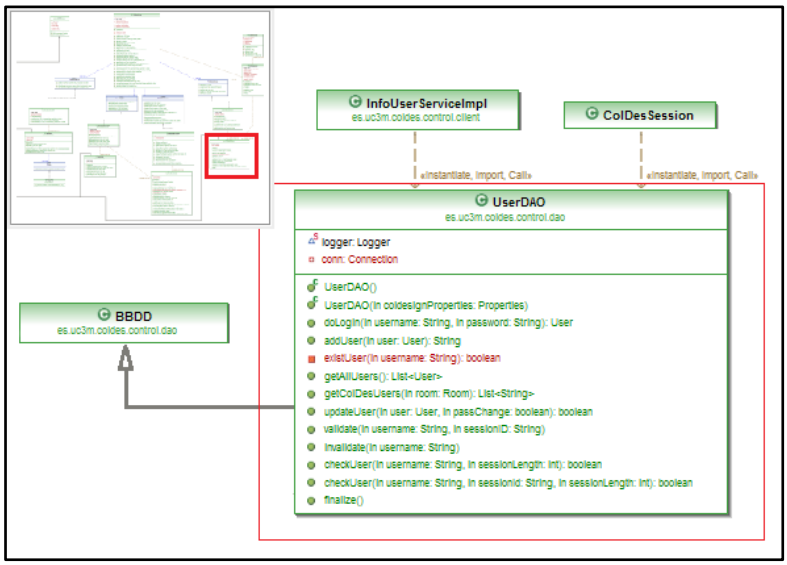


Figura 128: DAO de Usuarios

Este *DAO* es usado desde el *service* de usuarios, **InfoUserServiceImpl**, y por la clase que modela la sesión dentro del sistema, **CoDesSession**, de tal forma que para cada una de las acciones que el usuario realice dentro del sistema se pueda verificar si el usuario aún tiene la sesión activa.

Nombre	UserDAO.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	conn	*Connection*	Conexión a la base de datos.	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	UserDAO	*Constructor*	-	
	Descripción			
	Constructor por defecto en el que se leen las propiedades de la base de datos del fichero de propiedades *coldes.properties.*			
	Nombre	*Tipo*	*Argumentos*	
	UserDAO	*Contructor*	*Properties* ColDesProperties	
	Descripción			
	Constructor que recibe ya las distintas propiedades de la conexión a base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	doLogin	*User*	*String* user *String* password	
	Descripción			
	Verifica si el nombre de usuario y la password usados para intentar acceder a la aplicación son correctos. Esta función diferencia mayúsculas de minúsculas a la hora de hacer la comprobación. Para verificar la password del usuario se le aplica una función hash, que se compara con lo almacenado en la base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	addUser	*String*	*User* user	
	Descripción			

	Añade un nuevo usuario al sistema. Todos los datos del usuario se almacenan en claro en la base de datos salvo la contraseña, a la cual se le aplica una función hash y se almacena el resultado de esta operación.		
	Nombre	*Tipo*	*Argumentos*
	existUser	*Boolean*	*String* username
	Descripción		
	Verifica si un usuario dado existe mediante el nombre de usuario, campo que es usado como clave en la tabla de usuarios, por lo que no puede haber dos usuarios con el mismo nombre de usuario.		
	Nombre	*Tipo*	*Argumentos*
	getAllUsers	*List <User>*	-
	Descripción		
	Obtiene todos los usuarios registrados en el sistema.		
	Nombre	*Tipo*	*Argumentos*
	getColDesUsers	*List <String>*	*Room* room
	Descripción		
	Obtiene los usuarios del sistema que no se encuentran registrados en una sala dada. Este método es usado para saber cuáles son los usuarios a los cuales se les puede mandar una invitación puesto que no están registrados en la sala.		
	Nombre	*Tipo*	*Argumentos*
	updateUser	*Boolean*	*User* user *boolean* passChange
	Descripción		
	Actualiza los datos personales de un usuario, a excepción del nombre de usuario.		
Métodos Sesión	*Nombre*	*Tipo*	*Argumentos*
	validate	*void*	*String* username *String* sessionID
	Descripción		
	Función que valida la sesión de un usuario, actualizando el id de sesión, la fecha de última conexión y la de última operación.		
	Nombre	*Tipo*	*Argumentos*
	invalidate	*void*	*String* username
	Descripción		
	Invalida la sesión de un usuario dado poniendo a *null* el valor actual de su id de sesión.		
	Nombre	*Tipo*	*Argumentos*
	checkUser	*Boolean*	*String* username *String* sessionID *int* sessionLength
	Descripción		
	Función encargada de verificar si la sesión de un usuario es válida en el momento actual.		
	Nombre	*Tipo*	*Argumentos*
	finalize	*void*	-
	Descripción		
	Función encargada de finalizar la conexión del DAO con la base de datos.		

Tabla 96: Clase UserDAO

- DAO de salas

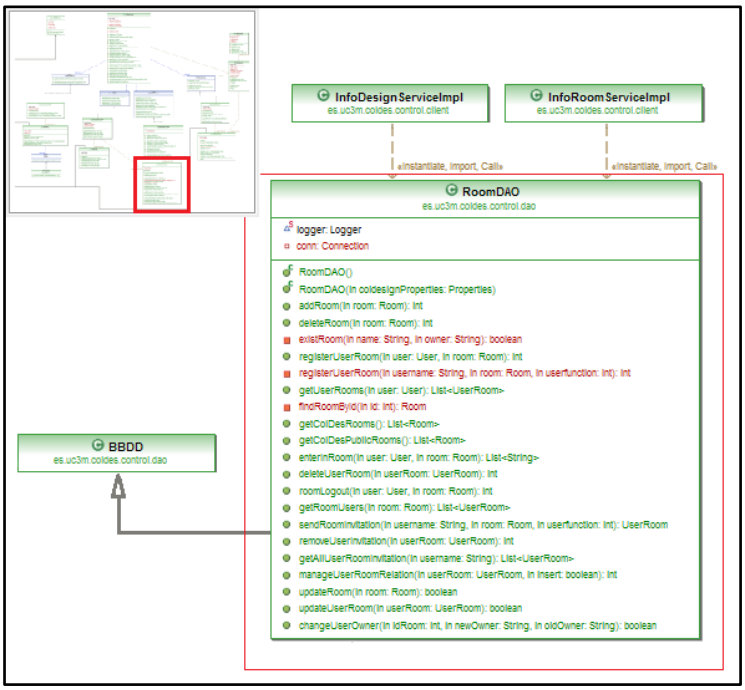


Figura 129: DAO de Usuarios

Este *DAO* es usado desde el *service* de salas, **InfoRoomServiceImpl**, y por otro parte para asociar diseños a la sala también es usado por el *service* de diseños, **InfoDesignServiceImpl**.

Nombre	RoomDAO.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	conn	*Connection*	Conexión a la base de datos.	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	RoomDAO	*Constructor*	-	
	Descripción			
	Constructor por defecto en el que se leen las propiedades de la base de datos del fichero de propiedades *coldes.properties.*			
	Nombre	*Tipo*	*Argumentos*	
	RoomDAO	*Contructor*	*Properties* ColDesProperties	
	Descripción			
	Constructor que recibe ya las distintas propiedades de la conexión a base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	addRoom	*int*	*Room* room	
	Descripción			
	Añade una nueva sala al sistema, y crea la relación usuario sala para el creador de la sala, propietario de la misma.			
	Nombre	*Tipo*	*Argumentos*	
	deleteRoom	*int*	*Room* room	
	Descripción			

	Borra una sala del sistema. El borrado de la sala conlleva también el borrado de todos los elementos que hacen referencia a esta sala. El borrado se controla en base de datos de tal forma de que al borrar una sala se borra o actualiza en cascada todos aquellos elementos que tengan el id de la sala como clave foránea.		
	Nombre	*Tipo*	*Argumentos*
	existRoom	*Boolean*	*String* name *String* owner
	Descripción		
	Verifica si para un usuario existe alguna sala suya que tenga un nombre especifico.		
	Nombre	*Tipo*	*Argumentos*
	registerUserRoom	*int*	*User* user *Room* room
	Descripción		
	Función mediante la cual se registra a un usuario en una sala como colaborador. Esta función se usa cuando un usuario se registra directamente en una sala.		
	Nombre	*Tipo*	*Argumentos*
	registerUserRoom	*int*	*User* user *Room* room *int* userfunction
	Descripción		
	Función mediante la cual se registra a un usuario en una sala con una función específica. Este método se usa cuando algún usuario del sistema es invitado a participar en una sala.		
	Nombre	*Tipo*	*Argumentos*
	getUserRooms	*List <UserRoom>*	*User* user
	Descripción		
	Obtiene todas las salas en las cuales está participando un usuario dado sea cual sea la función que este desempeña dentro de la misma.		
	Nombre	*Tipo*	*Argumentos*
	findRoomById	*Room*	*int* id
	Descripción		
	Busca una sala del sistema a partir de su id.		
	Nombre	*Tipo*	*Argumentos*
	getColDesRooms	*List <Room>*	-
	Descripción		
	Obtiene todas las salas que están registradas en el sistema.		
	Nombre	*Tipo*	*Argumentos*
	getColDesPublicRooms	*List <Room>*	-
	Descripción		
	Obtiene la lista de salas en las cuales está participando un usuario, así como la función que desempeña en cada una de las salas.		
	Nombre	*Tipo*	*Argumentos*
	getColDesRooms	*List <Room>*	-
	Descripción		
	Obtiene las distintas salas del sistema sin excepción.		
	Nombre	*Tipo*	*Argumentos*
	getColDesPublicRooms	*List <Room>*	-
	Descripción		

	Obtiene todas las salas del sistema que son de carácter público. Una sala de carácter público es visible para todos los usuarios, mientras las salas privadas son visibles únicamente para las personas que participan en dichas salas.		
	Nombre	*Tipo*	*Argumentos*
	enterInRoom	*List <String>*	*User* user *Room* room
	Descripción		
	Método al cual se recurre para saber que usuarios se encuentran en el momento actual participando dentro de la sala cuando un determinado usuario entra en la sala. Adicionalmente actualiza la relación entre el usuario y la sala, marcando el flag online a 1, para indicar que el usuario está participando en este momento en la sala.		
	Nombre	*Tipo*	*Argumentos*
	deleteUserRoom	*int*	*UserRoom* userRoom
	Descripción		
	Borra una relación entre un usuario y una sala.		
	Nombre	*Tipo*	*Argumentos*
	roomLogout	*int*	*User* user *Room* room
	Descripción		
	Función mediante la cual se controla la salida de un usuario de una determinada sala. Actualiza el flag online de la relación usuario-sala a 0.		
	Nombre	*Tipo*	*Argumentos*
	getRoomUsers	*List <UserRoom>*	*Room* room
	Descripción		
	Obtiene todas las relaciones usuario-sala de una determinada sala.		
	Nombre	*Tipo*	*Argumentos*
	sendRoomInvitation	*UserRoom*	*String* username *Room* room *int* userfunction
	Descripción		
	Envía una invitación a un usuario para participar en una determinada sala con una función establecida.		
	Nombre	*Tipo*	*Argumentos*
	removeUserInvitation	*int*	*UserRoom* userRoom
	Descripción		
	Obtiene todas las relaciones usuario-sala de una determinada sala.		
	Nombre	*Tipo*	*Argumentos*
	getAllUserRoomInvitation	*List <UserRoom>*	*String* username
	Descripción		
	Obtiene todas las invitaciones de un usuario determinado.		
	Nombre	*Tipo*	*Argumentos*
	manageUserRoomRelation	*int*	*UserRoom* userRoom *Boolean* insert
	Descripción		
	Función mediante la cual se gestionan las distintas acciones que un usuario puede realizar con las invitaciones que recibe. Primero borra la invitación que se está manejando, y posteriormente, si el usuario ha aceptado la invitación, se realiza el registro del usuario a la sala.		
	Nombre	*Tipo*	*Argumentos*
	updateRoom	*Boolean*	*Room* room
	Descripción		

	Actualiza los datos de una determinada sala.		
	Nombre	*Tipo*	*Argumentos*
	updateUserRoom	*Boolean*	*UserRoom* userRoom
	Descripción		
	Actualiza una relación usuario-sala determinada.		
	Nombre	*Tipo*	*Argumentos*
	changeUserOwner	*Boolean*	*int* idRoom *String* newOwner *String* oldOwner
	Descripción		
	Cambia el propietario de una determinada sala. Para ello actualiza la función que desempeñaba el nuevo propietario a propietario y cambia la función del propietario a moderador.		

Tabla 97: Clase RoomDAO

- DAO de pinceles

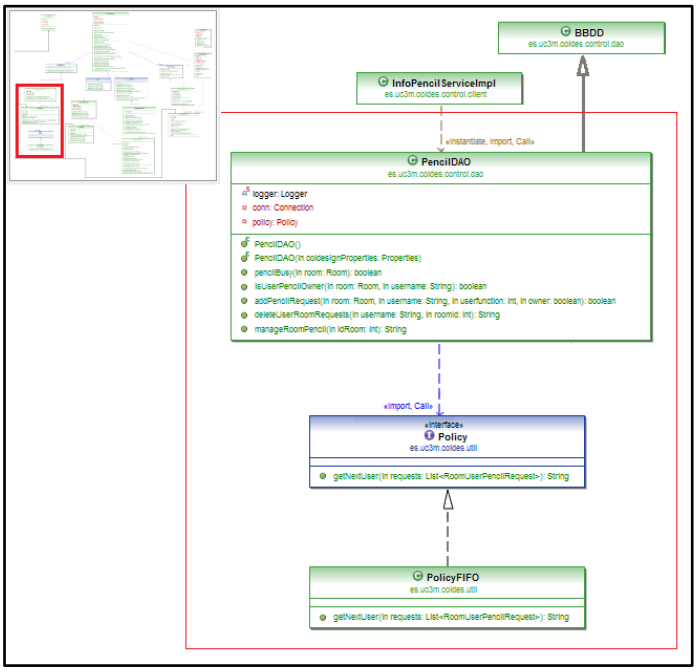


Figura 130: DAO de Pinceles

Este *DAO* es usado desde el *service* de pinceles, **InfoPencilServiceImpl**. Este *DAO* es el encargado de gestionar todo lo referente a las distintas peticiones de pinceles de las salas y de asignar el pincel siguiendo una política de asignación de pinceles que comentaremos más adelante.

Nombre	PencilDAO.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	conn	*Connection*	Conexión a la base de datos.	
	policy	*Policy*	Política mediante la cual se administran las distintas peticiones de pinceles en el sistema. La política que se usa por defecto es **FIFO** (*First Input First Output*)	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	PencilDAO	*Constructor*	-	
	Descripción			
	Constructor por defecto en el que se leen las propiedades de la base de datos del fichero de propiedades *coldes.properties.*			
	Nombre	*Tipo*	*Argumentos*	
	PencilDAO	*Contructor*	*Properties* ColDesProperties	
	Descripción			
	Constructor que recibe ya las distintas propiedades de la conexión a base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	pencilBusy	*Boolean*	*Room* room *String* username *int* userfunction	
	Descripción			
	Obtiene si el pincel de una sala dada esta siendo usado o no.			
	Nombre	*Tipo*	*Argumentos*	
	isUserPencilOwner	*Boolean*	*Room* room *String* username	
	Descripción			
	Verifica si un usuario es el dueño del pincel de una determinada sala.			
	Nombre	*Tipo*	*Argumentos*	
	addPencilRequest	*Boolean*	*Room* room *String* username *int* userfunction	
	Descripción			
	Función mediante la cual se añade una nueva petición de pincel de la sala al sistema.			
	Nombre	*Tipo*	*Argumentos*	
	deleteUserRoomRequests	*String*	*String* username *int* roomID	
	Descripción			
	Función que borra las peticiones de un usuario de una sala. Este método se invocara cuando un usuario salga del sistema o de una determinada sala únicamente. En caso de salir del sistema, se llamara a este método con los ids de las distintas salas en las que el usuario se encontraba participando. Esta función se encarga además de devolver el nombre de usuario del siguiente usuario propietario del pincel de la sala, o null en caso de no haber un siguiente.			
	Nombre	*Tipo*	*Argumentos*	
	manageRoomPencil	*String*	*int* idRoom	
	Descripción			

	Se manejan las distintas peticiones de una sala dada para obtener el siguiente propietario del pincel y actualizar los datos de las peticiones actuales del sistema. Usa la clase Policy para asignar el siguiente propietario del pincel de la clase. Por defecto, la PolicyFIFO.

Tabla 98: Clase PencilDAO

La asignación de los pinceles de las distintas salas se realiza entre las distintas peticiones que el sistema ha procesado dentro de la sala. Para seleccionar el siguiente propietario del pincel dentro de cada sala, se debe usar una clase que implemente los métodos definidos en la interfaz *Policy*. El proyecto se ha desarrollado siguiendo por defecto una política FIFO, en la que la primera petición que se realiza es la primera petición que se atiende.

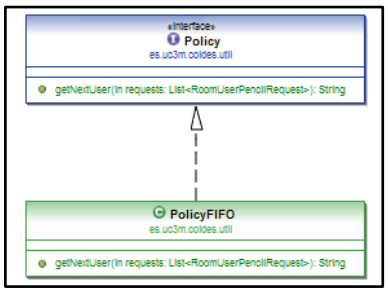


Figura 131: Policy

Nombre	Policy.java / PolicyFIFO.java		**Tipo**	Interfaz /Clase
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	getNextUser	*String*	*List<RoomUserPencilRequest>* request	
	Descripción			
	Función que se ha de implementar encargada de seleccionar la siguiente petición de una lista de peticiones. En este caso la siguiente petición es la primera de la lista que se le pasa, o *null* en caso de no existir peticiones.			

Tabla 99: Interfaz/Clase Policy.java/PolicyFIFO.java

- DAO de diseños

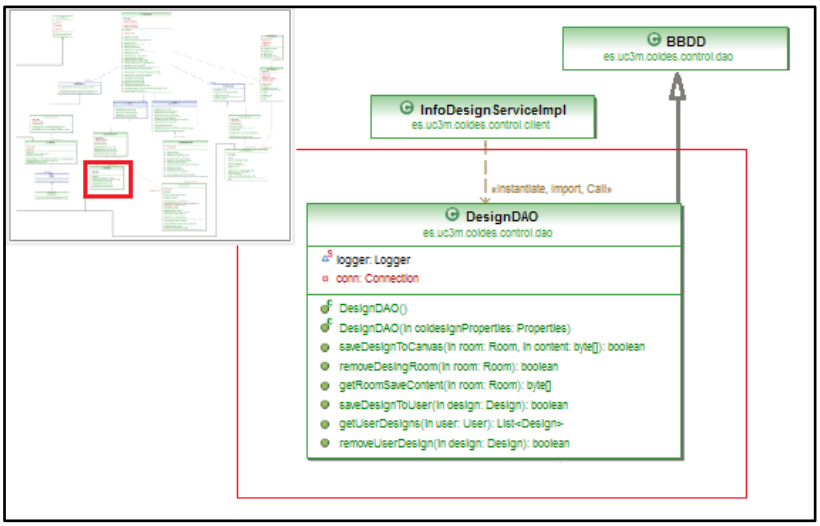


Figura 132: DAO de Diseños

Este *DAO* es usado desde el *service* de diseños, **InfoDesignServiceImpl**.

Nombre	DesignDAO.java		**Tipo**	Clase
Atributos	*Nombre*	*Tipo*	*Descripción*	
	conn	*Connection*	Conexión a la base de datos.	
Métodos	*Nombre*	*Tipo*	*Argumentos*	
	DesignDAO	*Constructor*	-	
	Descripción			
	Constructor por defecto en el que se leen las propiedades de la base de datos del fichero de propiedades *coldes.properties.*			
	Nombre	*Tipo*	*Argumentos*	
	DesignDAO	*Contructor*	*Properties* ColDesProperties	
	Descripción			
	Constructor que recibe ya las distintas propiedades de la conexión a base de datos.			
	Nombre	*Tipo*	*Argumentos*	
	saveDesignToCanvas	*Boolean*	*Room* room *byte[]* content	
	Descripción			
	Asigna el contenido del lienzo de una sala. Cada sala admite por el momento un único diseño a lo largo del tiempo, por ello, se busca si la sala tiene algún diseño asociado, en caso afirmativo, se actualiza el valor del contenido, y en caso contrario se inserta un nuevo registro.			
	Nombre	*Tipo*	*Argumentos*	
	removeDesignRoom	*Boolean*	*Room* room	
	Descripción			
	Desvincula el diseño perteneciente a una sala. Para ello borra el vínculo que se crea al asociar un diseño a una sala.			
	Nombre	*Tipo*	*Argumentos*	
	getRoomSaveContent	*byte[]*	*Room* room	
	Descripción			

	Obtiene el diseño asociado a una sala dada.		
	Nombre	*Tipo*	*Argumentos*
	saveDesignToUser	*Boolean*	*Design* design
	Descripción		
	Asocia un diseño a un usuario que ha decidido guardarlo. El propio objeto Design contiene el nombre de usuario.		
	Nombre	*Tipo*	*Argumentos*
	getUserDesigns	*List <Design>*	*User* user
	Descripción		
	Obtiene todos los diseños que ha guardado un usuario dado.		
	Nombre	*Tipo*	*Argumentos*
	removeUserDesign	*Boolean*	*Design* design
	Descripción		
	Desvincula un diseño de un usuario dado. Para ello borra el vínculo que se crea entre el diseño y el usuario en el momento de su vinculación.		

Tabla 100: Clase DesignDAO

Anexo VI. Scripts SQL de la base de datos

En este apartado se adjunta el código sql de creación de los distintos elementos del modelo del sistema, en sintaxis de la base de datos MySQL.

```
-- MySQL Administrator dump 1.4
--
-- ------------------------------------------------------
-- Server version 5.1.52-community

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

--
-- Create schema coldesbd
--
CREATE DATABASE IF NOT EXISTS coldesbd;
USE coldesbd;
--
-- Definition of table `room`
--
DROP TABLE IF EXISTS `room`;
CREATE TABLE `room` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `roomname` varchar(45) NOT NULL,
  `roomdescription` text NOT NULL,
  `owner` varchar(45) NOT NULL,
  `private` tinyint(1) NOT NULL,
  `participationtype` int(10) unsigned NOT NULL,
  `status` int(10) unsigned NOT NULL,
  `creationdate` datetime NOT NULL,
  `modificationdate` datetime NOT NULL,
  PRIMARY KEY (`id`),
  KEY `ownerIndex` (`owner`),
  CONSTRAINT `roomOwerFK` FOREIGN KEY (`owner`) REFERENCES `user`
(`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `room`
--
/*!40000 ALTER TABLE `room` DISABLE KEYS */;
/*!40000 ALTER TABLE `room` ENABLE KEYS */;


--
-- Definition of table `roomdesign`
--
DROP TABLE IF EXISTS `roomdesign`;
```

```
CREATE TABLE `roomdesign` (
  `id_room` int(10) unsigned NOT NULL,
  `designcontent` longtext NOT NULL,
  PRIMARY KEY (`id_room`),
  CONSTRAINT `FK_roomdesignIdRoom` FOREIGN KEY (`id_room`) REFERENCES
`room` (`id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `roomdesign`
--

/*!40000 ALTER TABLE `roomdesign` DISABLE KEYS */;
/*!40000 ALTER TABLE `roomdesign` ENABLE KEYS */;


--
-- Definition of table `roomuser`
--
DROP TABLE IF EXISTS `roomuser`;
CREATE TABLE `roomuser` (
  `id_room` int(10) unsigned NOT NULL,
  `username` varchar(45) NOT NULL,
  `userfunction` int(10) unsigned NOT NULL,
  `online` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id_room`,`username`),
  KEY `RoomId` (`id_room`),
  KEY `UsernameId` (`username`),
  CONSTRAINT `FK_id_room` FOREIGN KEY (`id_room`) REFERENCES `room`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FK_username` FOREIGN KEY (`username`) REFERENCES `user`
(`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `roomuser`
--

/*!40000 ALTER TABLE `roomuser` DISABLE KEYS */;
/*!40000 ALTER TABLE `roomuser` ENABLE KEYS */;


--
-- Definition of table `roomuserinvitation`
--
DROP TABLE IF EXISTS `roomuserinvitation`;
CREATE TABLE `roomuserinvitation` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `username` varchar(45) NOT NULL,
  `id_room` int(10) unsigned NOT NULL,
  `userfunction` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `Index_Username` (`username`),
  KEY `Index_IdRoom` (`id_room`),
  CONSTRAINT `FK_invroom` FOREIGN KEY (`id_room`) REFERENCES `room`
(`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FK_invusername` FOREIGN KEY (`username`) REFERENCES
`user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--
-- Dumping data for table `roomuserinvitation`
--

/*!40000 ALTER TABLE `roomuserinvitation` DISABLE KEYS */;
/*!40000 ALTER TABLE `roomuserinvitation` ENABLE KEYS */;


--
-- Definition of table `roomuserpencil`
--
DROP TABLE IF EXISTS `roomuserpencil`;
CREATE TABLE `roomuserpencil` (
  `id_room` int(10) unsigned NOT NULL,
  `username` varchar(45) NOT NULL,
  `userfunction` int(10) unsigned NOT NULL,
  `requesttime` datetime NOT NULL,
  `pencilowner` tinyint(1) NOT NULL,
  PRIMARY KEY (`id_room`,`username`),
  KEY `FK_pencilUsername` (`username`),
  CONSTRAINT `FK_pencilIdRoom` FOREIGN KEY (`id_room`) REFERENCES
`room` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FK_pencilUsername` FOREIGN KEY (`username`) REFERENCES
`user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `roomuserpencil`
--

/*!40000 ALTER TABLE `roomuserpencil` DISABLE KEYS */;
/*!40000 ALTER TABLE `roomuserpencil` ENABLE KEYS */;


--
-- Definition of table `user`
--
DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `username` varchar(45) NOT NULL,
  `password` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  `surname1` varchar(45) NOT NULL,
  `surname2` varchar(45) NOT NULL,
  `email` varchar(100) NOT NULL,
  `admin` tinyint(1) NOT NULL,
  `designer` tinyint(1) NOT NULL,
  `active` tinyint(1) NOT NULL,
  `sessionid` varchar(45) DEFAULT NULL,
  `lastlogin` datetime DEFAULT NULL,
  `lastoperation` datetime DEFAULT NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `user`
--
```

```
/*!40000 ALTER TABLE `user` DISABLE KEYS */;
INSERT INTO `user`
(`username`,`password`,`name`,`surname1`,`surname2`,`email`,`admin`,`d
esigner`,`active`,`sessionid`,`lastlogin`,`lastoperation`) VALUES
('admin','d033e22ae348aeb5660fc2140aec35850c4da997','Administrador','A
dministrador','Administrador','admin@coldes.es',1,1,1,NULL,'2011-11-01
10:45:29','2011-11-01 10:45:46');
/*!40000 ALTER TABLE `user` ENABLE KEYS */;


--
-- Definition of table `userdesign`
--
DROP TABLE IF EXISTS `userdesign`;
CREATE TABLE `userdesign` (
  `username` varchar(45) NOT NULL,
  `designname` varchar(45) NOT NULL,
  `designcontent` longtext NOT NULL,
  PRIMARY KEY (`username`,`designname`) USING BTREE,
  CONSTRAINT `FK_userdesignUsername` FOREIGN KEY (`username`)
REFERENCES `user` (`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Dumping data for table `userdesign`
--
```

Anexo VII. Evaluación de los usuarios

Incidencias

- Botón ‘Cancelar’ para el inicio de sesión que permita borrar los datos.
- Poner el registro como un enlace y no del mismo tamaño que el botón de acceder
- Pondría el botón de crear sala arriba. Más visible, quizá antes del buscar
- Privada (al crear una sala)¿Qué significa exactamente que sea privada?¿puedo luego invitar a personas y hacerla pública?
- Si una sala es privada, entendiendo como tal que es solo para mi, debería desaparecer la opción de seleccionar el modo de participación, puesto que solo yo voy a utilizarla.
- Cambiaría el nombre del botón de crear una sala (ahora ‘Registrar’) por ‘Crear’
- Añadir una columna a la tabla de las salas que indique el tipo de participación que tiene asignada esa sala.
- Cuando se invita a un usuario y se tiene que indicar el tipo de función, incluiría una breve descripción de los “derechos” que tiene cada usuario.
- Permitir eliminar usuarios (ADMIN)
- Añadir iconos que indiquen que puedes editar los datos de los usuarios (ADMIN)
- Cerrar automáticamente la pantalla de aceptar o rechazar invitación una realizada la acción
- Permitir ver la descripción o información de la sala antes de aceptar o rechazar una invitación
- Cambiar icono de eliminar relación con una sala, y pondría el icono de un enlace roto (como el del link roto)
- Añadir estado de la invitación al usuario que la realiza con la intención de evitar enviar varias invitaciones (se reciben varias invitaciones)
- Mejorar búsqueda, si no se pone nada, ni si quiera un espacio en blanco, mantener la lista completa.
- ¿Sobre qué se hace la búsqueda? ¿Sobre el nombre de la sala, el nombre del usuario? ¿Cualquier campo?

- No veo la diferencia entre el botón de ‘Restaurar’ y el de actualizar
- Si un usuario sale de la sala mientras tiene el pincel, debe perderlo automáticamente
- Al usuario que tiene el pincel se lo indica, pero no al resto de usuarios
- Hacer el log más grande (en mi resolución solo se ve la última línea)
- Hacer que el chat se pueda poner en otro sitio.
- Mejorar la disposición de los botones e incluso los iconos
- Poner por defecto el lápiz (libre) para empezar a escribir, sin necesidad de que el usuario tenga que seleccionar como quiere pintar.
- Mejorar el refresco cuando entra un usuario nuevo
- Salvar en mis diseños - botón para finalizar lo llamaría ‘Guardar’ no ‘Registrar’
- ***IMPORTANTE*** - Sección mis diseños. Guardo los diseños y no puedo recuperarlos!!
- No tiene que poder loguearse dos veces el mismo usuario
- A veces falla al salir de la sesión
- A revisar: activar usuario

Ocurrencias:

- ¿Repositorio común a una sala en la que se puedan subir otros diseños como ejemplo?
- Estado de la sala, si está en edición, si está cerrada...
- La opción “Usuarios” únicamente activa el botón de “Mis Datos”. El plural da a entender que podrás hacerse cosas con varios usuarios cuando no es así, es sólo sobre mis datos. Pondría “Consulta usuario”.
- En “Administración”. La acción se llama “Administrar salas” pero luego en la pestaña pone “Gestión de salas”. Usar nombre distintos incumple dos reglas de Nielsen.
- En “Administración”. En la “Gestión de salas” no está muy claro porque puede hacer selección de filas si la gestión se hace pulsando sobre los iconos correspondientes.

- En “Administración”. En la “Gestión de salas” el texto del tooltip que aparece al posicionarse sobre el icono de usuario es poco descriptivo.
- Aunque no lo cambiaría, es un poco confuso que determinadas acciones se hagan pulsando sobre iconos de la propia lista, en otros sean botones asociados a la pantalla, en otros enlaces...
- En “Administración”. La acción se llama “Administrar usuarios” pero luego en la pestaña pone “Gestión de usuarios”. Usar nombre distintos incumple dos reglas de Nielsen.
- En “Administración”. En la “Gestión de usuarios”, ¿por qué la tipografía del nombre de usuario es distinta?
- En “Administración”. En la “Gestión de usuarios”, el doble click sobre una fila sirve para mostrar editar la información del usuario; sin embargo, en la “gestión de salas” no hacía nada. Debería tener un comportamiento homogéneo.
- En ninguno de los formularios de la parte de administración se indica que campos son obligatorios.
- En “Salas”. En la opción de “Buscar salas” el tratamiento de las listas es también distinto: se puede hacer doble click pero sirve para mostrar un menú emergente en el que indicar la acción a realizar.
- En “Salas”. En la ventana de “Mis salas” de “eliminar relación” no es nada ilustrativo de la tarea.

Anexo VIII. Pruebas de Aceptación del Usuario

A continuación se presentan las distintas incidencias – mejoras – problemas de concepto detectadas por los usuarios. Todas ellas seguirán la siguiente nomenclatura:

- Identificador: valor que diferenciará unívocamente a cada elemento.
  - Incidencia (INC-XXX).
  - Mejora (MEJ-XXX).
  - Problema de concepto (CON-XXX).
- Tipo: Indica si el registro es incidencia, mejora o concepto.
- Descripción: explicación breve del problema.
- Elemento afectado: Indica a que elemento de la aplicación hace referencia el problema. Hemos dividido los distintos elementos en:
  - Funcionalidad
  - Interfaz
  - Lenguaje

Identificador	Tipo	Descripción	Elemento afectado
INC-001	Incidencia	No queda claro que un usuario administrador pueda editar los datos personales de un usuario.	*Interfaz*
INC-002	Incidencia	Cerrar la ventana emergente de invitaciones cuando el usuario acepte o decline alguna de las invitaciones.	*Funcionalidad*
INC-003	Incidencia	Evitar que se puedan enviar varias invitaciones de una misma sala a un determinado usuario	*Funcionalidad*
INC-004	Incidencia	El botón de restaurar y el de refresco realizan la misma función.	*Interfaz*
INC-005	Incidencia	Si un usuario que tiene el pincel dentro de una sala sale de la misma, automáticamente ha de perder el pincel.	*Funcionalidad*
INC-006	Incidencia	Dentro de una sala, es visual si el propio usuario tiene el pincel o no, pero cualquier otro usuario de la sala que no tenga el pincel no es consciente de que usuario tiene el pincel en cada determinado momento.	*Funcionalidad*
INC-007	Incidencia	Un usuario no puede autenticarse dos veces.	*Funcionalidad*
INC-008	Incidencia	A veces falla al salir de la sesión	*Funcionalidad*
INC-009	Incidencia	Evitar que en la pantalla de	*Interfaz*

Identificador	Tipo	Descripción	Elemento afectado
		Administración de Salas, el usuario pueda seleccionar cada una de las filas de la sala dado que la gestión se realiza mediante los botones definidos en el lado izquierdo de los datos de la sala.	
INC-010	Incidencia	En los distintos formularios de la aplicación no se especifica que campos son obligatorios.	*Interfaz*

Tabla 101: Incidencias

Identificador	Tipo	Descripción	Elemento afectado
MEJ-001	Mejora	Botón 'Cancelar' en la pantalla de autenticación que permita borrar los datos.	*Interfaz*
MEJ-002	Mejora	Cambiar el botón de registro para diferenciarlo del botón de acceso a la aplicación.	*Interfaz*
MEJ-003	Mejora	Cambiar la localización del botón de crear una nueva sala para hacerlo más visible en la ventana de "*Mis salas*".	*Interfaz*
MEJ-004	Mejora	En la creación de una nueva sala, cambiar el nombre del formulario de "Registro *de nueva sala*" a "*Crear nueva sala*".	*Lenguaje*
MEJ-005	Mejora	Añadir una nueva columna en las ventanas en las que se representan las salas (búsqueda o personales) que indique el tipo de participación de cada una de ellas.	*Interfaz*
MEJ-006	Mejora	A la hora de invitar a un usuario nuevo a la sala, se ve necesaria una breve descripción de lo que conlleva cada una de las funciones de los usuarios dentro de las salas.	*Interfaz*
MEJ-007	Mejora	Permitir a los usuarios administradores borrar usuarios creados en el sistema.	*Funcionalidad*
MEJ-008	Mejora	Permitir al usuario ver la descripción de la sala antes de aceptar o rechazar la invitación.	*Funcionalidad*
MEJ-009	Mejora	Cambiar el icono de "Dejar de participar" de la ventana de "Mis Salas" para que sea más explicativo de su funcionalidad.	*Interfaz*
MEJ-010	Mejora	Mejorar la búsqueda de usuarios y salas para permitir que en caso de no escribir nada o espacios, se mantenga la lista completa de elementos.	*Funcionalidad*
MEJ-011	Mejora	Mejorar la disposición de los botones y de sus iconos dentro de las salas.	*Interfaz*
MEJ-012	Mejora	En el momento que un usuario se hace con el pincel dentro de una sala, se ha de seleccionar por defecto que la herramienta de dibujo sea el lápiz.	*Funcionalidad*
MEJ-013	Mejora	Cambiar el formulario de guardado de	*Lenguaje*

Identificador	Tipo	Descripción	Elemento afectado
		diseños para usuarios de "*Registrar*" a "*Guardar*".	
MEJ-014	Mejora	No se puede acceder a los diseños de un usuario nada más que desde una sala. Sería necesario crear una nueva ventana que permitiese al usuario gestionar y pre visualizar sus propios diseños.	*Funcionalidad*
MEJ-015	Mejora	Cambiar la activación del usuario. Cuando un usuario nuevo se registre que este activo directamente.	*Funcionalidad*
MEJ-016	Mejora	Incluir en las salas un estado que indique si esta en edición, cerrada, …	*Funcionalidad*
MEJ-017	Mejora	Cambiar el menú de "*Usuarios*" por "*Usuario*" para evitar errores de concepto.	*Lenguaje*
MEJ-018	Mejora	Cambiar el nombre de la pestaña de "*Gestión de Salas*" por "*Administrar Salas*" para evitar incumplir reglas de Nielsen.	*Lenguaje*
MEJ-019	Mejora	El texto del tooltip del botón de administración de usuarios dentro de Administración de Salas es poco descriptivo.	*Lenguaje*
MEJ-020	Mejora	Cambiar el nombre de la pestaña de "*Gestión de Usuarios*" por "*Administrar Usuarios*" para evitar incumplir reglas de Nielsen.	*Lenguaje*
MEJ-021	Mejora	Cambiar la forma de editar los datos de los usuarios en Administración de Usuarios a algo semejante a como se realiza en Administración de Salas, para mantener de esta forma una funcionalidad homogénea.	*Interfaz*
MEJ-022	Mejora	El tooltip del botón para dejar de participar en una sala de la ventana de "*Mis Salas*" es poco ilustrativo.	*Lenguaje*

Tabla 102: Mejoras

Identificador	Tipo	Descripción	Elemento afectado
CON-001	Concepto	No queda claro el concepto de sala privada.	-
CON-002	Concepto	Al realizar una búsqueda de usuarios o salas, no se sabe sobre que campos de estos elementos se realiza dicha búsqueda.	-
CON-003	Concepto	En Administración de Usuarios, la tipografía del nombre de usuario es distinta, ¿Por qué?	-

Tabla 103: Problemas de concepto

A continuación se describen las acciones realizadas para solventar los problemas descritos en las anteriores tablas:

Identificador	Solución adoptada
INC-001	Se cambia la pantalla de administración de usuarios asemejándola a la pantalla de administración de salas. Se incluye un botón mediante el cual el administrador puede acceder directamente a los datos del usuario para modificarlos.
INC-002	Se cambia la forma de actuar de esta ventana para que cuando un usuario acepte o rechace una invitación se cierre directamente.
INC-003	Se cambia la consulta que realiza sobre base de datos para verificar que usuarios no están en la sala (usuarios potenciales a ser invitados), y se agrega la clausula where de la consulta el que el usuario al que se ha invitado no tenga ya una invitación activa.
INC-004	Se elimina el botón de restaurar para evitar funcionalidad duplicada.
INC-005	El usuario para perder el pincel ha de salir de la sala mediante el botón "x" de la pestaña de dicha sala, o mediante el botón salir de la aplicación. Si el usuario sale de forma inesperada del sistema, esta acción no se puede controlar.
INC-006	Se cambia el chat de la sala para que refleje en todo momento que usuario tiene el pincel. El usuario que tenga el pincel aparecerá en rojo en los nombres de los usuarios que están participando en la sala, visibles en el componente chat de la sala.
INC-007	El usuario que se autentica por segunda vez estando ya en el sistema, anula la sesión anterior. Esto puede provocar los fallos descritos en la INC-008.
INC-008	Tiene relación con la INC-007
INC-009	Se evita que las tablas se puedan seleccionar, pero se deja acceso a los botones de gestión.
INC-010	Se agrega el carácter "*" a los campos obligatorios de los formularios de registro del sistema. De forma adicional, si se envía el formulario al servidor sin haber complementado los campos obligatorios el sistema indicara al usuario que ha de rellenar dichos campos marcándoselos con un reborde rojo.
MEJ-001	Se agrega un botón "*Cancelar*" encargado de resetear el formulario de acceso.
MEJ-002	Se cambia de posición y de formato el botón "*Registrar*" de la aplicación.
MEJ-003	Se mueve el botón "*Crear Sala*" a la parte superior de la pantalla para hacerlo más visible.
MEJ-004	Se cambia el nombre del formulario de "Registro *de nueva sala*" a "*Crear nueva sala*".
MEJ-005	Se añade una nueva columna en las ventanas en las que se representan las salas (búsqueda o personales) que indica el tipo de participación de cada una de ellas.
MEJ-006	A la hora de invitar a un nuevo usuario a la sala, se coloca un tooltip sobre cada una de las funciones con una breve explicación de lo que cada una de ellas permite hacer al usuario dentro de la sala.
MEJ-007	Se agrega la funcionalidad en la parte de Administración de usuarios de borrar usuarios existentes del sistema. Este borrado supondrá también el borrado de las relaciones del usuario con las salas del sistema, el borrado de las salas de las cuales es propietario, y el borrado de los diseños asociados al usuario.
MEJ-008	Se añade un textbox dentro de la ventana de invitaciones de usuario en el que se muestra la descripción de la sala asociada a la invitación que el usuario

Identificador	Solución adoptada
	seleccione.
MEJ-009	Se cambia el icono de "Dejar de participar" de la ventana de "Mis Salas" para que sea más explicativo de su funcionalidad.
MEJ-010	Mejorar la búsqueda de usuarios y salas para permitir que en caso de no escribir nada o espacios, se mantenga la lista completa de elementos.
MEJ-011	Mejorar la disposición de los botones y de sus iconos dentro de las salas.
MEJ-012	Se hace que en el momento que un usuario coge el pincel de una sala, se selecciona directamente la herramienta de dibujo de lápiz.
MEJ-013	Se cambia el formulario de guardado de diseños para usuarios de "*Registrar*" a "*Guardar*".
MEJ-014	Se agrega una nueva ventana en la cual el usuario podrá ver los distintos diseños que tiene almacenados y de esta forma pre visualizarlos y gestionarlos. A este nuevo menú se accederá mediante el menú de "*Usuario*" y de ahí desde el submenú "*Mis diseño*s".
MEJ-015	A la hora de registrar un nuevo usuario, el flag active del mismo irá siempre como activo.
MEJ-016	Incluir en las salas un estado que indique si esta en edición, cerrada, …
MEJ-017	Se cambia el menú de "*Usuarios*" por "*Usuario*" para evitar errores de concepto.
MEJ-018	Se cambia el nombre de la pestaña de "*Gestión de Salas*" por "*Administrar Salas*".
MEJ-019	Se cambia el texto del tooltip del botón de administración de usuarios dentro de Administración de Salas.
MEJ-020	Se cambia el nombre de la pestaña de "*Gestión de Usuarios*" por "*Administrar Usuarios*".
MEJ-021	Se cambia la forma de trabajar en la ventana de administración de usuarios para que la funcionalidad corresponda a la que sigue la pantalla de administración de salas. Se agregan unos submenús por registro de usuario mediante el cual se puede acceder a los datos de los usuarios.
MEJ-022	Se cambia El tooltip del botón para dejar de participar en una sala de la ventana de "*Mis Salas*".
CON-001	Se explica a los usuarios el concepto de sala privada.
CON-002	Al realizar una búsqueda de usuarios o salas, no se sabe sobre que campos de estos elementos se realiza dicha búsqueda.
CON-003	Por una lado se cambia la tipografía del nombre de usuario para mantenerla homogénea con el resto de campos. De forma adicional se explica a los usuarios porque el nombre del usuario viene en un color rojo o verde. Rojo el usuario esta desactivado, verde el usuario esta activo.

Tabla 104: Solución de problemas